# Guide to Informix Enterprise Replication

Version 9.2
September 1999
Part No. 000-6208

Published by Informix® Press

Informix Corporation
4100 Bohannon Drive
Menlo Park, CA 94025-1032

© 1999 Informix Corporation. All rights reserved. The following are trademarks of Informix Corporation or its affiliates, one or more of which may be registered in the United States or other jurisdictions:

Answers OnLine™; C-ISAM®; Client SDK™; DataBlade®; Data Director™; Decision Frontier™; Dynamic Scalable Architecture™; Dynamic Server™; Dynamic Server™, Developer Edition™; Dynamic Server™ with Advanced Decision Support Option™; Dynamic Server™ with Extended Parallel Option™; Dynamic Server™ with MetaCube®; Dynamic Server™ with Universal Data Option™; Dynamic Server™ with Web Integration Option™; Dynamic Server™, Workgroup Edition™; Dynamic Virtual Machine™; Enterprise Decision Server™; Formation™; Formation Architect™; Formation Flow Engine™; Gold Mine Data Access®; IIF.2000™; i.Reach™; i.Sell™; Illustra®; Informix®; Informix® 4GL; Informix® InquireSM; Informix® Internet Foundation.2000™; InformixLink®; Informix® Red Brick® Decision Server™; Informix Session Proxy™; Informix® Vista™; InfoShelf™; Interforum™; I-Spy™; Mediazation™; MetaCube®; NewEra™; ON-Bar™; OnLine Dynamic Server™; OnLine/Secure Dynamic Server™; OpenCase®; Orca™; PaVER™; Red Brick® and Design; Red Brick® Data Mine™; Red Brick® Mine Builder™; Red Brick® Decisionscape™; Red Brick® Ready™; Red Brick Systems®; Regency Support®; Rely on Red BrickSM; RISQL®; Solution DesignSM; STARindex™; STARjoin™; SuperView®; TARGETindex™; TARGETjoin™; The Data Warehouse Company®; The one with the smartest data wins.™; The world is being digitized. We're indexing it.SM; Universal Data Warehouse Blueprint™; Universal Database Components™; Universal Web Connect™; ViewPoint®; Visionary™; Web Integration Suite™. The Informix logo is registered with the United States Patent and Trademark Office. The DataBlade logo is registered with the United States Patent and Trademark Office.

Documentation Team: Linda Briscoe, Mary Kraemer, Jennifer Leland, Judith Sherwood

GOVERNMENT LICENSE RIGHTS

Software and documentation acquired by or for the US Government are provided with rights as follows: (1) if for civilian agency use, with rights as restricted by vendor's standard license, as prescribed in FAR 12.212; (2) if for Dept. of Defense use, with rights as restricted by vendor's standard license, unless superseded by a negotiated vendor license, as prescribed in DFARS 227.7202. Any whole or partial reproduction of software or documentation marked with this legend must reproduce this legend.

# Table of Contents

## Introduction

In This Introduction . . . . . 3
About This Manual . . . . . 3
Types of Users . . . . . 4
Software Dependencies . . . . . 4
Assumptions About Your Locale . . . . . 5
Demonstration Databases . . . . . 5
New Features . . . . . 6
Documentation Conventions . . . . . 7
Typographical Conventions . . . . . 8
Icon Conventions . . . . . 8
Command-Line Conventions . . . . . 10
Sample-Code Conventions . . . . . 12
Additional Documentation . . . . . 13
On-Line Manuals . . . . . 13
Printed Manuals . . . . . 14
On-Line Help . . . . . 14
Error Message Documentation . . . . . 14
Documentation Notes, Release Notes, Machine Notes . . . . . 15
Related Reading . . . . . 16
Compliance with Industry Standards . . . . . 16
Informix Welcomes Your Comments . . . . . 16

# Section I Introducing Informix Enterprise Replication

## Chapter 1 Data-Replication Overview


In This Chapter . . . . . 1-3
Data-Replication Types . . . . . 1-3
Synchronous Data Replication . . . . . 1-3
Asynchronous Data Replication. . . . . 1-4
Data-Replication Capture Mechanisms . . . . . 1-5
Trigger-Based Data Capture . . . . . 1-6
Trigger-Based Transaction Capture . . . . . 1-6
Log-Based Data Capture . . . . . 1-8
Informix Enterprise Replication Features . . . . . 1-8
High Performance . . . . . 1-9
High Availability . . . . . 1-9
Consistent Information Delivery . . . . . 1-10
Flexible Architecture . . . . . 1-10
Easy, Centralized Administration . . . . . 1-11
How Enterprise Replication Replicates Data . . . . . 1-12


## Chapter 2 Enterprise Replication Features


In This Chapter . . . . . 2-3
Enterprise Replication Topologies . . . . . 2-3
Topology Terminology . . . . . 2-4
Replication Topology Types . . . . . 2-5
Enterprise Replication Objects . . . . . 2-8
Enterprise Replication Server . . . . . 2-8
Replicate. . . . . 2-9
Participant . . . . . 2-12
Replicate Group . . . . . 2-12
Catalog . . . . . 2-14
Replication Types . . . . . 2-14
Primary-Target Replication . . . . . 2-15
Update-Anywhere Replication . . . . . 2-15
Conflict Resolution. . . . . 2-15
Conflict-Resolution Rules . . . . . 2-16
Resolution Scope . . . . . 2-16
Replication Data . . . . . 2-17
Evaluating Data for Replication . . . . . 2-17
Distributing Data. . . . . 2-19
Applying Replicated Data. . . . . 2-19

## Chapter 3 Replication Examples


In This Chapter . . . . . 3-3
Environment for the Replication Examples . . . . . 3-3
Terminology . . . . . 3-5
Primary-Target Example Using the CLU . . . . . 3-6
Preparing for a Primary-Target Replication . . . . . 3-6
Cause a Replication . . . . . 3-8
Do Not Cause a Replication . . . . . 3-9
Update-Anywhere Example Using the CLU . . . . . 3-9
Preparing for an Update-Anywhere Replication . . . . . 3-9
Cause a Replication . . . . . 3-12
Hierarchy Example Using the CLU . . . . . 3-13
Preparing for a Hierarchy Example . . . . . 3-14
Defining a Hierarchy . . . . . 3-14
Primary-Target Example Using ERM . . . . . 3-16
Removing Previously Defined Replication Servers. . . . . 3-17
Defining the First Replication Server . . . . . 3-18
Defining a Replication Server on italy . . . . . 3-24
Creating the Primary-Target Replicate . . . . . 3-26
Viewing the Replicate. . . . . 3-33
Activating the Replicate . . . . . 3-34
Cause a Replication . . . . . 3-36
Do Not Cause a Replication . . . . . 3-36
Update-Anywhere Example Using ERM . . . . . 3-37
Changing to Expert Mode . . . . . 3-37
Creating an Update-Anywhere Replicate . . . . . 3-37
Viewing the Replicate. . . . . 3-44
Defining a Replication Server on japan . . . . . 3-45
Adding a Participant to repl4 . . . . . 3-46
Activating the Replicate . . . . . 3-48
Cause a Replication . . . . . 3-49
Hierarchy Example Using ERM . . . . . 3-49
Deleting the Previously-Defined Servers . . . . . 3-49
Defining the Nonroot Replication Servers. . . . . 3-49
Defining a Leaf Replication Server . . . . . 3-50
Displaying the Hierarchy . . . . . 3-50
Viewing the Server Properties . . . . . 3-51

# Section II Preparing for Enterprise Replication

## Chapter 4 Selecting Enterprise-Replication Systems


In This Chapter . . . . . 4-3
Primary-Target Replication System . . . . . 4-3
Data Dissemination . . . . . 4-4
Data Consolidation . . . . . 4-5
Workload Partitions . . . . . 4-6
Workflow Replication . . . . . 4-7
Factors in Primary-Target Systems . . . . . 4-7
Selecting Data for Primary-Target Replication . . . . . 4-8
Administering the Replication System . . . . . 4-8
Planning for Capacity . . . . . 4-8
Planning for High Availability . . . . . 4-8
Update-Anywhere Replication System . . . . . 4-9
Facts to Consider . . . . . 4-9


## Chapter 5 Setting Up Your Replication Environment


In This Chapter . . . . . 5-3
Operational Limitations . . . . . 5-3
SQL Statement Use . . . . . 5-3
Forbidden SQL Statements . . . . . 5-4
Limited SQL Statements . . . . . 5-4
Permitted SQL Statements . . . . . 5-4
Capacity Planning . . . . . 5-5
Logical-Log Records . . . . . 5-5
Send and Receive Message Queues . . . . . 5-6
Delete Tables . . . . . 5-7
Spooling Directories . . . . . 5-7
Configuration Parameters . . . . . 5-8
Enterprise Replication Threads . . . . . 5-8
Environmental Considerations . . . . . 5-9
Time Synchronization . . . . . 5-9
Transaction-Processing Effect . . . . . 5-10
Enterprise Replication Data Types . . . . . 5-11
Replicating on Heterogeneous Hardware . . . . . 5-11
Replicating Simple Large Objects . . . . . 5-11
Replicating User-Defined Data Types . . . . . 5-15
Replicating Smart Large Objects . . . . . 5-15
Using GLS with Enterprise Replication . . . . . 5-15

**Chapter 6 Preparing Data for Replication**

In This Chapter . . . . . 6-3
Preparing Consistent Data . . . . . 6-3
Shadow Columns . . . . . 6-4
Using the High-Performance Loader . . . . . 6-4
Using the onunload and onload Utilities . . . . . 6-5
Using the dbexport and dbimport Utilities . . . . . 6-5
Using the UNLOAD and LOAD statements . . . . . 6-5
Blocking Replication . . . . . 6-6
Data Preparation Examples . . . . . 6-8
Adding a New Participant to an Existing Replicate . . . . 6-8
Beginning Work Without Replication . . . . . 6-9

Section III Replicating Data

**Chapter 7 Enterprise Replication Processes**

In This Chapter . . . . . 7-3
Capturing Transactions . . . . . 7-3
Evaluating Data for Replication . . . . . 7-4
Evaluating Time . . . . . 7-4
Evaluating Images of a Row . . . . . 7-5
Evaluating Replication Examples. . . . . 7-7
Evaluating Primary-Key Updates . . . . . 7-10
Receiving and Sending Data from the Message Queue . . . 7-11
Applying Data for Distribution . . . . . 7-12
Choosing Conflict-Resolution Rules and Scopes . . . . . 7-13
Time-Stamp Conflict-Resolution Rule . . . . . 7-15
Using Cascading Deletes and Triggers . . . . . 7-20
Using Cascading Deletes . . . . . 7-21
Using Triggers . . . . . 7-21

## Chapter 8 Enterprise Replication Menus


In This Chapter . . . . . 8-5
General Information . . . . . 8-6
Preparing to Use the Replication Manager . . . . . 8-6
Opening the Replication Manager . . . . . 8-6
Understanding Server Terminology . . . . . 8-7
Using Modes . . . . . 8-7
Choosing Ellipses . . . . . 8-8
Viewing Long Identifiers . . . . . 8-8
The Replication Manager Main Window . . . . . 8-10
Using the Global Catalog Icon . . . . . 8-10
Using the What's This Icon . . . . . 8-11
The File Menu . . . . . 8-11
Creating Scripts . . . . . 8-11
Monitoring Replication Events . . . . . 8-11
The Replicate Menu . . . . . 8-12
Managing Replicates and Participants . . . . . 8-13
Changing Replicate States. . . . . 8-14
Changing the Primary Participant . . . . . 8-17
Viewing Replicate Properties. . . . . 8-17
Group Menu . . . . . 8-20
Defining a Replicate Group . . . . . 8-20
Modifying Groups . . . . . 8-20
Changing the State of a Group . . . . . 8-22
Viewing Replicate Group Properties . . . . . 8-24
Server Menu . . . . . 8-25
Suspending and Resuming a Database Server . . . . . 8-25
Finding or Changing the Server State . . . . . 8-26
Viewing the Replication Hierarchy. . . . . 8-27
Changing to Primary . . . . . 8-28
Changing to Target . . . . . 8-28
Starting and Stopping Enterprise Replication . . . . . 8-28
Specifying Privileges . . . . . 8-29
Removing a Database Server from Enterprise Replication . . . 8-30
Reloading the Global Catalog . . . . . 8-31
Viewing Database Server Properties . . . . . 8-31
View Menu . . . . . 8-32
By Replicate and By Server . . . . . 8-32
Toolbar and Status Bar . . . . . 8-32

Window Menu . . . . . 8-33
Nonexpert Mode . . . . . 8-33
Expert Mode . . . . . 8-34
List of Open Windows . . . . . 8-34
The Help Menu . . . . . 8-35
Help Topics . . . . . 8-35
Using Help . . . . . 8-37
Contacting Informix . . . . . 8-37
About Replication Manager . . . . . 8-37
Hierarchical View Menus . . . . . 8-38
Replication Scripting View Menus . . . . . 8-39

**Chapter 9 Enterprise Replication Manager Wizards**

In This Chapter . . . . . 9-3
Defining a Replicate . . . . . 9-4
Defining a Replicate Name . . . . . 9-4
Choosing a Conflict-Resolution Procedure . . . . . 9-5
Choosing Replicate Options . . . . . 9-6
Selecting the Replication Frequency . . . . . 9-7
Selecting Servers to Participate in the Replicate . . . . . 9-9
Specifying Identical or Non-Identical Participants . . . . . 9-10
Defining Participant Attributes . . . . . 9-12
Choosing the Columns to Replicate . . . . . 9-13
Reviewing Replicate Summary . . . . . 9-14
Adding a Participant . . . . . 9-15
Defining a Replicate Group . . . . . 9-15
Choosing a Group Name . . . . . 9-16
Choosing the Replication Interval for the Group . . . . . 9-16
Selecting Replicates for the Replicate Group . . . . . 9-17
Generic Configuration Wizard . . . . . 9-17
Choosing a Naming Scheme . . . . . 9-18
Describing the Replicates . . . . . 9-20
Describing the Participants . . . . . 9-21
Customizing the Generic Replicates . . . . . 9-23

**Chapter 10 Diagnosing Enterprise Replication**

In This Chapter . . . . . 10-3
Aborted-Transaction Spooling . . . . . 10-3
Preparing to Use ATS . . . . . 10-4
BYTE and TEXT Information in ATS Files . . . . . 10-8
Row-Information Spooling . . . . . 10-9
BYTE and TEXT Information in RIS Files . . . . . 10-11
Replication Event Monitor Messages . . . . . 10-14

**Chapter 11 Command-Line Utility**

In This Chapter . . . . . 11-3
List of Commands . . . . . 11-5
Command-Line Abbreviations . . . . . 11-6
Option Abbreviations . . . . . 11-6
Order of the Options . . . . . 11-7
Backslash Use . . . . . 11-8
Connect Option . . . . . 11-8
Participant . . . . . 11-9
Participant Modifier . . . . . 11-11
Database Server Groups . . . . . 11-12
Return Codes . . . . . 11-13
Frequency Options . . . . . 11-14
Example Using the Command-Line Utility . . . . . 11-16
Command-Line Utility Syntax . . . . . 11-19
cdr change group . . . . . 11-20
cdr change replicate . . . . . 11-22
cdr connect server . . . . . 11-24
cdr define group . . . . . 11-25
cdr define replicate . . . . . 11-27
cdr define server . . . . . 11-33
cdr delete group . . . . . 11-38
cdr delete replicate . . . . . 11-39
cdr delete server . . . . . 11-40
cdr disconnect server . . . . . 11-43
cdr error . . . . . 11-44
cdr list group . . . . . 11-47
cdr list replicate . . . . . 11-48
cdr list server . . . . . 11-50
cdr modify group . . . . . 11-53
cdr modify replicate . . . . . 11-55
cdr modify server . . . . . 11-59

cdr resume group . . . . . 11-61
cdr resume replicate . . . . . 11-62
cdr resume server . . . . . 11-63
cdr start . . . . . 11-64
cdr start group . . . . . 11-66
cdr start replicate . . . . . 11-67
cdr stop . . . . . 11-68
cdr stop group . . . . . 11-70
cdr stop replicate . . . . . 11-71
cdr suspend group. . . . . 11-73
cdr suspend replicate . . . . . 11-75
cdr suspend server. . . . . 11-77

**Chapter 12 API Functions**

In This Chapter . . . . . 12-3
Basic Information for APIs . . . . . 12-3
Database Server Groups . . . . . 12-3
Time Specifications . . . . . 12-5
Identifiers . . . . . 12-5
API Structures . . . . . 12-5
Severe-Error Table . . . . . 12-7
Enterprise Replication APIs . . . . . 12-8
cdr_connect() . . . . . 12-10
cdr_define_group() . . . . . 12-11
cdr_define_repl() . . . . . 12-14
cdr_define_serv() . . . . . 12-18
cdr_delete_group() . . . . . 12-20
cdr_delete_repl() . . . . . 12-21
cdr_delete_serv() . . . . . 12-22
cdr_error_reviewed() . . . . . 12-23
cdr_modify_group() . . . . . 12-24
cdr_modify_repl() . . . . . 12-25
cdr_modify_replmode() . . . . . 12-27
cdr_modify_serv() . . . . . 12-28
cdr_modify_servmode() . . . . . 12-29
cdr_move_errortab() . . . . . 12-30
cdr_participate_group() . . . . . 12-31
cdr_participate_repl(). . . . . 12-32
cdr_prune_errors(). . . . . 12-33
cdr_prune_single_error() . . . . . 12-34
cdr_resume_group() . . . . . 12-35
cdr_resume_repl() . . . . . 12-36

cdr_resume_serv() . . . . . 12-37
cdr_start_cdr(). . . . . 12-38
cdr_start_group(). . . . . 12-39
cdr_start_repl() . . . . . 12-40
cdr_stop_cdr(). . . . . 12-41
cdr_stop_group(). . . . . 12-42
cdr_stop_repl() . . . . . 12-43
cdr_suspend_group() . . . . . 12-44
cdr_suspend_repl() . . . . . 12-45
cdr_suspend_serv() . . . . . 12-46
Example . . . . . 12-47

## Section IV Appendixes

**Appendix A Configuration Parameters**

**Appendix B SMI Tables for Enterprise Replication**

**Appendix C Return Codes**

**Appendix D Fine-Tuning Enterprise Replication**

**Appendix E SQLHOSTS Registry Key**

**Appendix F Enterprise Replication onstat -g Options**

# Introduction

In This Introduction . . . . . 3

About This Manual . . . . . 3
Types of Users . . . . . 4
Software Dependencies . . . . . 4
Using Enterprise Replication Manager . . . . . 4
Assumptions About Your Locale . . . . . 5
Demonstration Databases . . . . . 5

New Features . . . . . 6
Extensibility Enhancements . . . . . 6
Performance Improvements . . . . . 6
Special Features . . . . . 6

Documentation Conventions . . . . . 7
Typographical Conventions . . . . . 8
Icon Conventions . . . . . 8
Comment Icons . . . . . 9
Feature, Product, and Platform Icons . . . . . 9
Command-Line Conventions . . . . . 10
How to Read a Command-Line Diagram . . . . . 12
Sample-Code Conventions . . . . . 12

Additional Documentation . . . . . 13
On-Line Manuals . . . . . 13
Printed Manuals . . . . . 14
On-Line Help . . . . . 14
Error Message Documentation . . . . . 14
Documentation Notes, Release Notes, Machine Notes . . . . . 15
Related Reading . . . . . 16

Compliance with Industry Standards. . . . . 16
Informix Welcomes Your Comments . . . . . 16

## In This Introduction

This Introduction provides an overview of the information in this manual and describes the conventions it uses.

## About This Manual

This manual contains information to help you understand the concepts of data replication, install Enterprise Replication, design your replication system, and administer and manage data replication throughout your enterprise.

Informix Enterprise Replication provides you with an efficient way to replicate data asynchronously throughout your open-systems enterprise. Enterprise Replication is a client/server application that operates on UNIX and Windows NT. The Enterprise Replication Manager (ERM), a Windows NT-based graphical user interface (GUI), helps you easily define, monitor, and control your replication system.

## Types of Users

This manual is for database server administrators.

This manual assumes that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with database server administration, operating-system administration, or network administration

If you have limited experience with relational databases, SQL, or your operating system, refer to your *Getting Started* manual for a list of supplementary titles.

## Software Dependencies

This manual assumes that you are using one of the following database servers:

- Informix Dynamic Server, Version 7.31
- Informix Dynamic Server 2000, Version 9.2

You also need to install Informix Connect. I-Connect is part of the database server product bundle, but it has a separate install procedure.

### *Using Enterprise Replication Manager*

WIN NT

To use Enterprise Replication Manager with Dynamic Server, Version 7.31, install Informix Enterprise Command Center (IECC). From IECC, choose **Tools→Enterprise Replication Manager**.

To use Enterprise Replication Manager with Dynamic Server, Version 9.2, install ERM and the proper connectivity tools. For information about preparing ERM for Version 9.2, refer to the documentation notes described in "Documentation Notes, Release Notes, Machine Notes" on page 15 of this chapter. ♦

## Assumptions About Your Locale

Informix products can support many languages, cultures, and code sets. All culture-specific information is brought together in a single environment, called a Global Language Support (GLS) locale.

This manual assumes that you use the U.S. 8859-1 English locale as the default locale. The default is **en_us.8859-1** (ISO 8859-1) on UNIX platforms or **en_us.1252** (Microsoft 1252) for Windows NT environments. This locale supports U.S. English format conventions for dates, times, and currency, and also supports the ISO 8859-1 or Microsoft 1252 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *Informix Guide to GLS Functionality*.

## Demonstration Databases

The DB-Access utility, which is provided with your Informix database server products, includes one or more of the following demonstration databases:

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in Informix manuals are based on the **stores_demo** database.
- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines.

For information about how to create and populate the demonstration databases, see the *DB-Access User's Manual*. For descriptions of the databases and their contents, see the *Informix Guide to SQL: Reference*.

The scripts that you use to install the demonstration databases reside in the **$INFORMIXDIR/bin** directory on UNIX platforms and in the **%INFORMIXDIR%\bin** directory in Windows environments.

# New Features

For a comprehensive list of new database server features, see the release notes. This section describes new features relevant to this manual. These features fall into the following areas:

- Extensibility enhancements
- Performance improvements
- Special features

## Extensibility Enhancements

This manual describes the following extensibility enhancements to Version 9.2 of Dynamic Server: limited support of user-defined and opaque data types.

## Performance Improvements

This manual describes the following performance improvements for Enterprise Replication in Version 9.2 of Dynamic Server:

- Improvements in managing the storage queues
- Faster processing for large transactions
- Better management to avoid long transactions
- Improvements in the log reader

## Special Features

This manual describes the following special features of Version 9.2 of Dynamic Server:

- Long identifiers
  - 128-character identifier
  - 32-character user names
- Updates to the Enterprise Replication GUI and command-line interface
- Hierarchical routing; direct connections no longer required

- Additional support and performance enhancements for database servers with intermittent connections
- Reduce global catalog available
- Enhance GLS to replicate multiple locales within a single replication environment
- command-line utility
- Scripting view in the replication manager GUI

## Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other volumes in the documentation set.

The following conventions are discussed:

- Typographical conventions
- Icon conventions
- Command-line conventions
- Sample-code conventions
- Screen-illustration conventions

## Typographical Conventions

This manual uses the following conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

| Convention | Meaning |
|---|---|
| KEYWORD | All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font. |
| *italics*<br>***italics***<br>*italics* | Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics. |
| **boldface**<br>***boldface*** | Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface. |
| monospace<br>*monospace* | Information that the product displays and information that you enter appear in a monospace typeface. |
| KEYSTROKE | Keys that you are to press appear in uppercase letters in a sans serif font. |
| ♦ | This symbol indicates the end of one or more product- or platform-specific paragraphs. |
| → | This symbol indicates a menu item. For example, "Choose **Tools→Options**" means choose the **Options** item from the **Tools** menu. |

***Tip:*** *When you are instructed to "enter" characters or to "execute" a command, immediately press* RETURN *after the entry. When you are instructed to "type" the text or to "press" other keys, no* RETURN *is required.*

## Icon Conventions

Throughout the documentation, you will find text that is identified by several different types of icons. This section describes these icons.

## Comment Icons

Comment icons identify three types of information, as the following table describes. This information always appears in italics.

| Icon | Label | Description |
|---|---|---|
| | ***Warning:*** | Identifies paragraphs that contain vital instructions, cautions, or critical information |
| | ***Important:*** | Identifies paragraphs that contain significant information about the feature or operation that is being described |
| | ***Tip:*** | Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described |

## Feature, Product, and Platform Icons

Feature, product, and platform icons identify paragraphs that contain feature-specific, product-specific, or platform-specific information.

| Icon | Description |
|---|---|
| GLS | Identifies information that relates to the Informix Global Language Support (GLS) feature |
| UNIX | Identifies information that is specific to UNIX platforms |
| WIN NT | Identifies information that is specific to the Windows NT environment |

These icons can apply to an entire section or to one or more paragraphs within a section. If an icon appears next to a section heading, the information that applies to the indicated feature, product, or platform ends at the next heading at the same or higher level. A ♦ symbol indicates the end of feature-, product-, or platform-specific information that appears within one or more paragraphs within a section.

## Command-Line Conventions

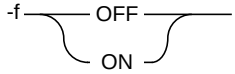
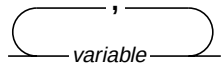
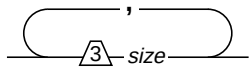
This section defines and illustrates the format of commands that are available in Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

Each diagram displays the sequences of required and optional elements that are valid in a command. A diagram begins at the upper-left corner with a command. It ends at the upper-right corner with a vertical line. Between these points, you can trace any path that does not stop or back up. Each path describes a valid form of the command. You must supply a value for words that are in italics.

You might encounter one or more of the following elements on a command-line path.

| Element | Description |
|---|---|
| command | This required element is usually the product name or other short word that invokes the product or calls the compiler or preprocessor script for a compiled Informix product. It might appear alone or precede one or more options. You must spell a command exactly as shown and use lowercase letters. |
| *variable* | A word in italics represents a value that you must supply, such as a database, file, or program name. A table following the diagram explains the value. |
| -flag | A flag is usually an abbreviation for a function, menu, or option name, or for a compiler or preprocessor argument. You must enter a flag exactly as shown, including the preceding hyphen. |

(1 of 2)

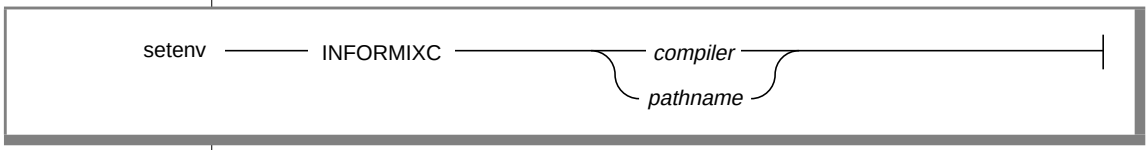
| Element | Description |
|---|---|
| .ext | A filename extension, such as **.sql** or **.cob**, might follow a variable that represents a filename. Type this extension exactly as shown, immediately after the name of the file. The extension might be optional in certain products. |
| ( . , ; + * - / ) | Punctuation and mathematical notations are literal symbols that you must enter exactly as shown. |
| ' ' | Single quotes are literal symbols that you must enter as shown. |
| Privileges p. 5-17<br>Privileges | A reference in a box represents a subdiagram. Imagine that the subdiagram is spliced into the main diagram at this point. When a page number is not specified, the subdiagram appears on the same page. |
| | A shaded option is the default action. |
| | Syntax within a pair of arrows indicates a subdiagram. |
| | The vertical line terminates the command. |
|  | A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.) |
|  | A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items. |
|  | A gate ( 3 ) on a path indicates that you can only use that path the indicated number of times, even if it is part of a larger loop. You can specify *size* no more than three times within this statement segment. |

(2 of 2)

### How to Read a Command-Line Diagram

Figure 1 shows a command-line diagram that uses some of the elements that are listed in the previous table.

**Figure 1**
*Example of a Command-Line Diagram*




To construct a command correctly, start at the top left with the command. Follow the diagram to the right, including the elements that you want. The elements in the diagram are case sensitive.

Figure 1 illustrates the following steps:

1. Type `setenv`.
2. Type `INFORMIXC`.
3. Supply either a compiler name or a pathname.

   After you choose *compiler* or *pathname*, you come to the terminator. Your command is complete.
4. Press RETURN to execute the command.

## Sample-Code Conventions

Examples of SQL code occur throughout this manual. Except where noted, the code is not specific to any single Informix application development tool. If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
    WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using DB-Access, you must delimit multiple statements with semicolons. If you are using an SQL application programming interface (API), you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement.

***Tip:*** *Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.*

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the manual for your product.

# Additional Documentation

For additional information, you might want to refer to the following types of documentation:

- On-line manuals
- Printed manuals
- On-line help
- Error message documentation
- Documentation notes, release notes, and machine notes
- Related reading

## On-Line Manuals

An Answers OnLine CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print on-line manuals, see the installation insert that accompanies Answers OnLine.

Informix on-line manuals are also available on the following Web site:

```
www.informix.com/answers
```

## Printed Manuals

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com. Please provide the following information when you place your order:

- The documentation that you need
- The quantity that you need
- Your name, address, and telephone number

WIN NT

## On-Line Help

Informix provides on-line help with each GUI that displays information about those interfaces and the functions that they perform. Use the help facilities that each GUI provides to display the on-line help.

## Error Message Documentation

Informix software products provide ASCII files that contain all of the Informix error messages and their corrective actions.

UNIX

To read error messages and corrective actions on UNIX, use one of the following utilities.

| Utility | Description |
|---|---|
| **finderr** | Displays error messages on line |
| **rofferr** | Formats error messages for printing |

♦

WIN NT

To read error messages and corrective actions in Windows environments, use the **Informix Find Error** utility. To display this utility, choose **Start→Programs→Informix** from the Task Bar. ♦

Instructions for using the preceding utilities are available in Answers OnLine. Answers OnLine also provides a listing of error messages and corrective actions in HTML format.

## Documentation Notes, Release Notes, Machine Notes

In addition to printed documentation, the following sections describe the on-line files that supplement the information in this manual. Please examine these files before you begin using your database server. They contain vital information about application and performance issues.

UNIX

The following on-line files appear in the **$INFORMIXDIR/release/en_us/0333** directory.

| On-Line File | Purpose |
|---|---|
| **EREPDOC_9.2** | The documentation notes file for your version of this manual describes topics that are not covered in the manual or that were modified since publication. |
| **SERVERS_9.2** | The release notes file describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds. |
| **IDS_9.2** | The machine notes file describes any special actions that you must take to configure and use Informix products on your computer. Machine notes are named for the product described. |

♦

WIN NT

The following items appear in the **Informix** folder. To display this folder, choose **Start→Programs→Informix** from the Task Bar.

| Program Group Item | Description |
|---|---|
| **Documentation Notes** | This item includes additions or corrections to manuals and information about features that might not be covered in the manuals or that were modified since publication. |
| **Release Notes** | This item describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds. |

Machine notes do not apply to Windows environments. ♦

## Related Reading

For a list of publications that provide an introduction to database servers and operating-system platforms, refer to your *Getting Started* manual.

# Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

# Informix Welcomes Your Comments

Let us know what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about any corrections or clarifications that you would find useful. Include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

Send electronic mail to us at the following address:

doc@informix.com

The **doc** alias is reserved exclusively for reporting errors and omissions in our documentation.

We appreciate your suggestions.

Section I

# Introducing Informix Enterprise Replication


**Chapter 1** **Data-Replication Overview**

**Chapter 2** **Enterprise Replication Features**

**Chapter 3** **Replication Examples**

Chapter

# 1

# Data-Replication Overview


In This Chapter . . . . . 1-3

Data-Replication Types . . . . . 1-3
- Synchronous Data Replication . . . . . 1-3
- Asynchronous Data Replication . . . . . 1-4

Data-Replication Capture Mechanisms . . . . . 1-5
- Trigger-Based Data Capture . . . . . 1-6
- Trigger-Based Transaction Capture . . . . . 1-6
- Log-Based Data Capture . . . . . 1-8

Informix Enterprise Replication Features . . . . . 1-8
- High Performance . . . . . 1-9
- High Availability . . . . . 1-9
- Consistent Information Delivery . . . . . 1-10
- Flexible Architecture . . . . . 1-10
  - Replication Models . . . . . 1-10
  - Network Topologies . . . . . 1-10
  - Data Types . . . . . 1-11
  - Operating Modes . . . . . 1-11
- Easy, Centralized Administration . . . . . 1-11

How Enterprise Replication Replicates Data . . . . . 1-12

## In This Chapter

Sharing corporate data is an important aspect of day-to-day business operations. Data replication generates and manages multiple copies of data at one or more sites, which allows an enterprise to share corporate data throughout its organization.

This chapter introduces the following information:

- Data-replication types
- Data-replication capture mechanisms
- Informix Enterprise Replication features

# Data-Replication Types

Data replication can be synchronous and asynchronous. Each data-replication type provides different capabilities and strengths. Informix database servers allow you to use either type of replication. Enterprise Replication provides asynchronous data replication.

## Synchronous Data Replication

In *synchronous* data replication, the replicated data is updated immediately when the source data is updated. Synchronous data replication uses the *two-phase commit* technology to protect data integrity. In a two-phase commit, a transaction is applied only if *all* interconnected distributed sites agree to accept the transaction.

Figure 1-1 illustrates the two-phase commit. The database server that initiates the transaction, Server A, asks each participating server whether it is ready to accept the transaction. In this figure, Server Z has not responded, therefore the transaction does not commit on *any* database server:

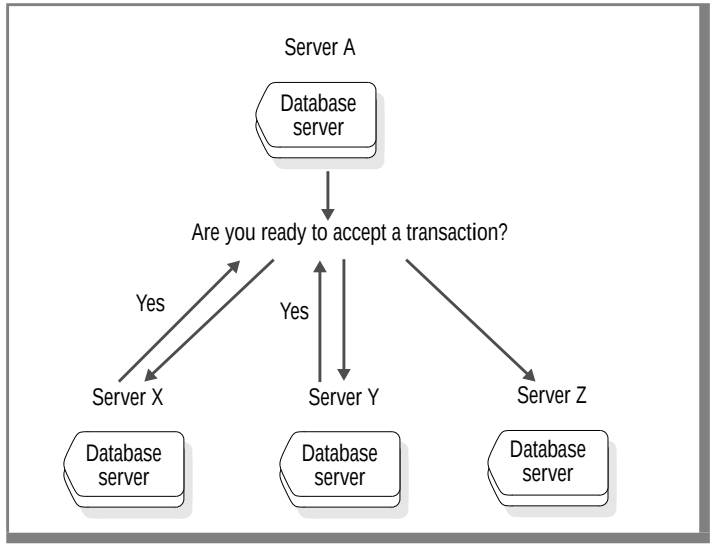



***Figure 1-1***
*Two-Phase Commit Technology*

Synchronous data replication is appropriate for applications that require immediate data synchronization. However, synchronous data replication requires that all hardware components and networks in the replication system be available at all times. It is difficult to guarantee the constant availability of all devices in a replication system because hardware components and networks fail. For more information about synchronous replication, refer to the discussion of two-phase commit in your *Administrator's Guide*.

## Asynchronous Data Replication

*Asynchronous* data replication updates the databases that reside at a replicated site after the primary database has committed the change. The delay to update the replicated-site databases can vary depending on the business application and user requirements. However, the data eventually synchronizes to the same value at all sites. The major benefit of this type of data replication is that if a particular database server fails, the replication process can continue and the original transaction is not directly affected by replication.

Asynchronous replication allows the following different replication models:

- Primary target

  All database changes originate at the primary database and are replicated to the target databases. Changes at the target databases are not replicated to the primary.

  The primary-target model can *distribute* information, using one primary database replicating to several secondary databases or *collect* information using several primary databases replicating to a single secondary.

- Update anywhere

  All databases have read and write capabilities. Updates are applied at all databases.

The update-anywhere model provides the greater challenge in asynchronous replication. For example, if a replication system contains three replication sites that all have read and write capabilities, update conflicts occur when the sites try to update the same data at the same time. Update conflicts must be detected and resolved so that the data elements eventually have the same value at every site.

Asynchronous replication is the more common type of data replication in open-system environments because it protects against the system and network failures.

## Data-Replication Capture Mechanisms

Asynchronous data replication updates the target databases after the primary database has committed the change. Several mechanisms can be used to capture data in asynchronous replication.The most frequently used data-replication capture mechanisms are:

- Trigger-based data capture
- Trigger-based transaction capture
- Log-based capture

## Trigger-Based Data Capture

A trigger is associated with a piece of data. When a change is made to the data item, the trigger can activate the replication process. The intent of this data-capture mechanism is to protect the referential integrity of the data. However, when a trigger begins to transfer modified data items, trigger-based data capture does not keep track of transaction information. Because trigger-based capture does not keep track of transaction information, customers and vendors must produce applications that ensure the referential integrity of the data.

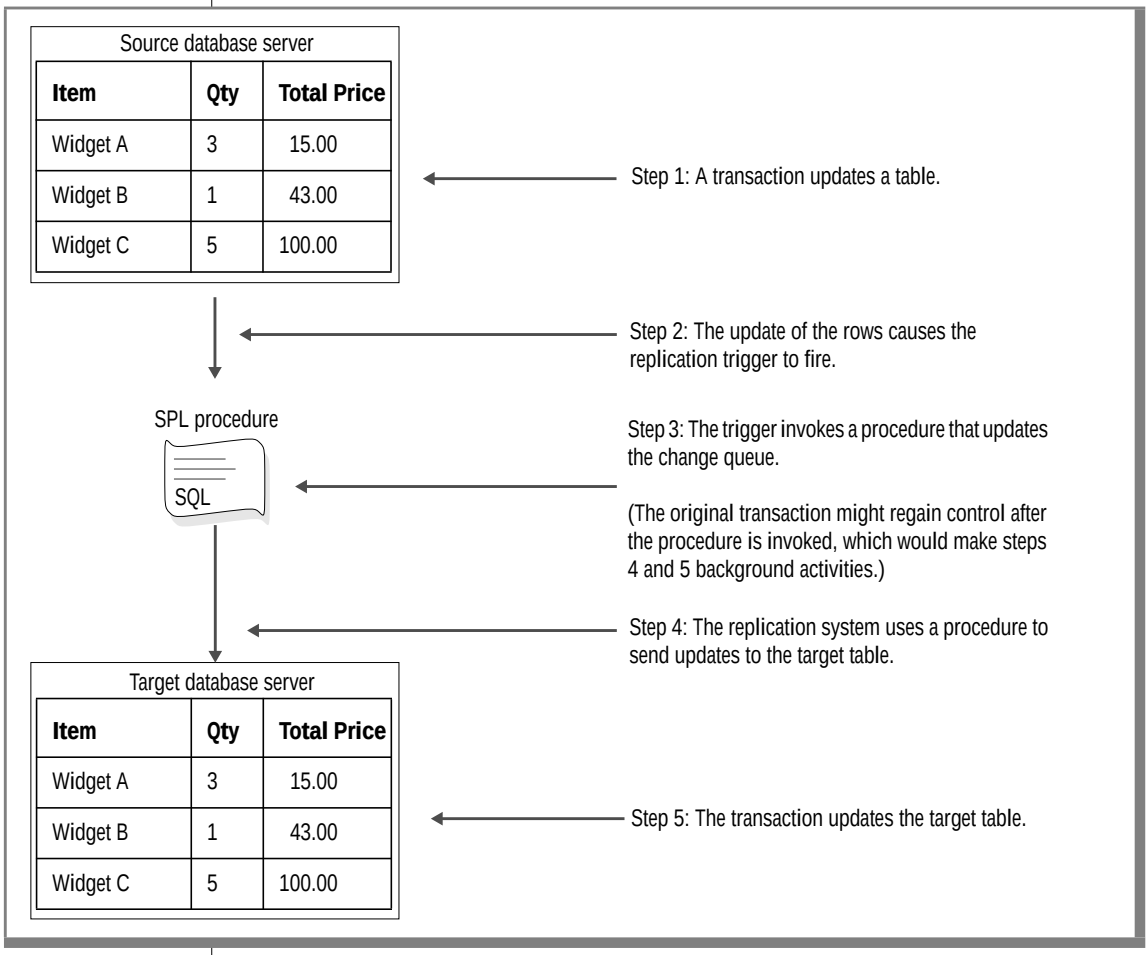
***Tip:*** *Triggers are used for many other purposes besides replication. For a broader discussion of triggers, refer to the "Informix Guide to Database Design and Implementation."*

## Trigger-Based Transaction Capture

Trigger-based transaction capture does not perform replication based on triggered data. Instead data changes are grouped into transactions. This data-capture mechanism assures the referential integrity of the data but adds additional work that can affect system performance.

In trigger-based transaction capture, a single transaction might trigger several replications if it modifies several tables because the triggers are associated with the table. In this case, the trigger receives the whole transaction, but the procedure that captures the data runs as a part of the original transaction, thus slowing down the original transaction. Figure 1-2 illustrates this type of replication.

***Figure 1-2***
*Trigger-Based Transaction Capture Affects the Performance of the Original Transaction*



## Log-Based Data Capture

Enterprise Replication uses *log-based data capture*. Log-based data capture takes changes from the logical log and does not compete with transactions for access to production tables. Log-based data-capture systems operate as part of the normal database-logging process and add minimum overhead to the system. However, not all log-based replication systems are the same. Log-based replication systems that use external database servers and processes offset the advantages of log-based transaction capture because of the additional data-transfer overhead.

An efficient log-based transaction-capture mechanism is an integral part of the database system.

To replicate data throughout an open-systems enterprise is much more complex than to copy data from one database server to another. In many situations, the most efficient and cost-effective data replication is an asynchronous replication that uses a log-based transaction-capture mechanism.

# Informix Enterprise Replication Features

Many data replication products use the following key elements to successfully replicate data throughout an open-system enterprise:

- High performance
- High availability
- Consistent information delivery
- Flexible architecture
- Easy, centralized administration

This section associates the key elements of data replication with Enterprise Replication features.

## High Performance

A replication system must not overly burden the source of the data, must use networks efficiently, and must use all resources efficiently.

Enterprise Replication uses a log-based transaction-capture mechanism that is integrated in the database architecture. The database server can implement this capture mechanism efficiently because the capture mechanism is internal to the database. In addition, all replication operations are performed in parallel, which further extends the performance of Enterprise Replication.

Enterprise Replication operates in a LAN, WAN, and combined LAN / WAN configuration across a range of network transport protocols. Enterprise Replication uses the standard database server communication facility to establish network connectivity among the Enterprise Replication servers involved in replication.

## High Availability

A replication system must be reliable and must not expose business operations to computer system failures.

Enterprise Replication implements asynchronous data replication. Asynchronous replication ensures that network and target database server outages are tolerated. Enterprise Replication also uses a delivery mechanism that stores data locally and then sends the data to the remote database server in two separate transactions. Target database servers acknowledge receipt of a message when the message is *safely stored*. A message is safely stored when it is either in a queue or in the target database.

In the event of a database server or network failure, the local database server can continue to service the local users.

## Consistent Information Delivery

A replication system must protect data integrity.

All Enterprise Replication transactions are stored in a reliable queue to maintain the consistency of transactions. Additionally, Enterprise Replication maintains the order of transactions when data is replicated. Maintaining the consistency and order of transactions ensures that if the same record is updated multiple times, the changes are applied to the remote databases in the same order as on the primary database and applied only once at the target database server.

If update conflicts occur, Enterprise Replication provides built-in automatic conflict detection and resolution. You can configure the way conflict resolution behaves to meet the needs of your database.

## Flexible Architecture

A replication system should not impose model or methodology restrictions on the enterprise. A replication system should allow replications based on specific business and application requirements.

### *Replication Models*

Enterprise Replication supports both primary-target and update-anywhere replication models.

### *Network Topologies*

Enterprise Replication supports the following network topologies:

- Fully interconnected

  Continuous connectivity between all participating database servers.
- Hierarchical tree

  A parent-child configuration that supports continuous and intermittent connectivity.
- Forest of trees

  Multiple hierarchical tree that connects at the root database servers.

For more information, see "Enterprise Replication Topologies" on page 2-3.

### Data Types

Enterprise Replication supports all built-in Informix data types across heterogeneous hardware. Enterprise Replication also supports the Global Language Support (GLS) feature, which allows Informix products to handle different languages, cultural conventions, and code sets.

Enterprise Replication provides limited support for user-defined data types.

### Operating Modes

The Enterprise Replication Manager (ERM) provides two operating modes: nonexpert and expert. The nonexpert operating mode supports only primary-target replication and requires that all participating database servers be fully interconnected. The expert operating mode supports all ownership models and hierarchical replication.

## Easy, Centralized Administration

Administrators must be able to easily manage all the distributed components of the replication system from a single point of control.

ERM, a Windows NT GUI, allows an administrator to define and manage the replication system. You do not need to shut down the entire replication system to add database servers to the replication system. For information about ERM, see Chapter 8, "Enterprise Replication Menus."

You can also use the command-line utility (CLU) to administer the replication system from your system command prompt. This utility is documented in Chapter 11, "Command-Line Utility."

An Enterprise Replication *global catalog* (a system database table) on each database server keeps track of Enterprise Replication configuration information. The global catalog maintains information such as replication definitions, Enterprise Replication servers, conflict detection and resolution rules, and their associated SPL procedures. Any changes to replication definitions are automatically sent to other database servers in the replication system.

# How Enterprise Replication Replicates Data

Before you can replicate data, you must declare a database server for replication and define replicates. Declaring a database server for replication creates a database that keeps information about all servers in the replication system and about all data items that should be replicated. Chapter 3, "Replication Examples," has simple examples of declaring replication servers and defining replicates. For additional information, refer to Chapter 5, "Setting Up Your Replication Environment."

After the replicates are defined, Enterprise Replication uses the following four major processes to replicate data:

- Capture transactions
- Evaluate data for replication
- Distribute data for replication
- Apply replicated data

Detailed information on each process is in Chapter 7, "Enterprise Replication Processes."

Chapter

# 2

# Enterprise Replication Features


In This Chapter . . . . . 2-3

Enterprise Replication Topologies . . . . . 2-3
- Topology Terminology . . . . . 2-4
- Replication Topology Types . . . . . 2-5
  - Fully Connected . . . . . 2-5
  - Hierarchical Tree . . . . . 2-6
  - Forest of Trees . . . . . 2-7

Enterprise Replication Objects . . . . . 2-8
- Enterprise Replication Server . . . . . 2-8
- Replicate . . . . . 2-9
  - Replicate Options. . . . . 2-9
  - Replicate State . . . . . 2-10
  - Conflict-Resolution Rules . . . . . 2-10
  - Replicate Example . . . . . 2-10
- Participant . . . . . 2-12
- Replicate Group . . . . . 2-12
  - Ease of Use . . . . . 2-13
  - Increased Performance . . . . . 2-13
- Catalog . . . . . 2-14

Replication Types . . . . . 2-14
- Primary-Target Replication. . . . . 2-15
  - One-to-Many Replication . . . . . 2-15
  - Many-to-One Replication . . . . . 2-15
- Update-Anywhere Replication . . . . . 2-15

Conflict Resolution . . . . . 2-15
- Conflict-Resolution Rules . . . . . 2-16
- Resolution Scope . . . . . 2-16

Replication Data . . . . . 2-17
Evaluating Data for Replication . . . . . 2-17
Evaluating Time . . . . . 2-17
Evaluating Images of a Row . . . . . 2-17
Processing in Parallel . . . . . 2-18
Distributing Data . . . . . 2-19
Applying Replicated Data . . . . . 2-19

## In This Chapter

This chapter introduces the following Enterprise Replication features:

- Replication topologies
- Replication objects
- Replication types
- Conflict resolution
- Replication Data

The next chapter, Chapter 3, "Replication Examples," presents simple replication examples that use some of the objects defined in this chapter.

# Enterprise Replication Topologies

Enterprise Replication *topology* describes connections that replication servers make to interact with each other. You must define the replication topology within the topology of the physical network, but the replication topology is not synonymous with the physical network topology. The order in which you define the database servers for replication and the options that you use determine the replication topology.

Enterprise Replication provides three replication topologies:

- Fully connected
- Hierarchical tree
- Forest of trees

For information on how to define replication servers, see "cdr define server" on page 11-33 and "Preparing to Use the Replication Manager" on page 8-6.

## Topology Terminology

Enterprise Replication uses the terms in the Figure 2-1 to describe replication topology.

***Figure 2-1***
*Replication Topology Terms*

| Term | Definition |
|---|---|
| Root server | An Enterprise Replication server that is the uppermost level in a hierarchically organized set of information. The root is the point from which database servers branch into a logical sequence. All root database servers within Enterprise Replication must be fully interconnected. |
| Nonroot server | An Enterprise Replication server that is not a root database server but that has a complete global catalog and is connected to its parent and to its children. |
| Tree | A data structure that contains database server(s) that are linked in a hierarchical manner. The topmost node is called the root. The root can have zero or more *child* database servers; the root is the *parent* database server to its children. |
| Parent-Child | A relationship between database servers in a tree data structure in which the parent is one step closer to the root than the child. |
| Leaf server | A database server that has a limited catalog and no descendants. |

A *root* server is fully connected to all other root servers. It has information about all other replication servers in its replication environment. Figure 2-2 on page 2-5 shows an environment with four root servers.

A *nonroot server* is similar to a root server except that it forwards all replicated messages for other root servers (and their children) through its parent. All nonroot servers are known to all root and other nonroot servers. A nonroot server can be a terminal point in a tree or it can be the parent for other servers. All root and nonroot servers are aware of all other servers in the replication environment.

A *leaf* server is a server with a sparse catalog. A leaf server has knowledge only of itself and its parent server. A leaf server is a terminal point in a tree; it cannot be a parent to another replication server.

## Replication Topology Types

The topology that you choose influences the types of replication that you can use. The next sections describe the topologies that Enterprise Replication supports.

### *Fully Connected*

*Fully-connected* replication topology indicates that all database servers connect to each other and that Enterprise Replication establishes and manages the connections. Replication messages are sent directly from one database server to another. No additional routing is necessary to deliver replication messages. Figure 2-2 shows a fully-connected replication topology. Each database server connects directly to every other database server in the replication environment.

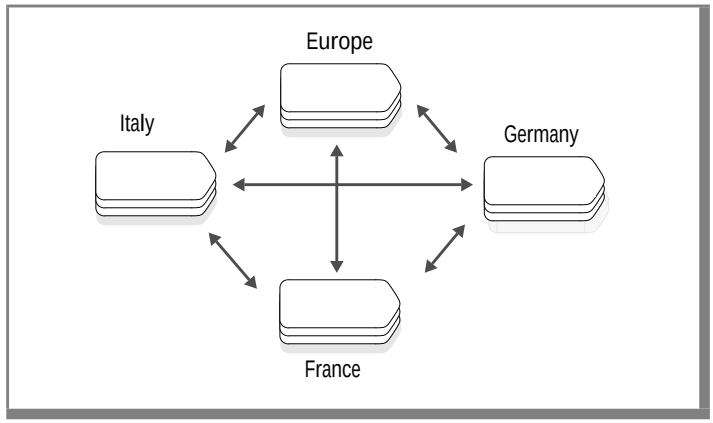



***Figure 2-2***
*Fully Connected Topology*

## Hierarchical Tree

A *hierarchical tree* consists of a root database server and one or more database servers organized into a tree topology. The tree contains only one root, which has no parent. Each database server within the tree references its parent. A database server that is not a parent is a leaf. Figure 2-3 illustrates a replication tree.

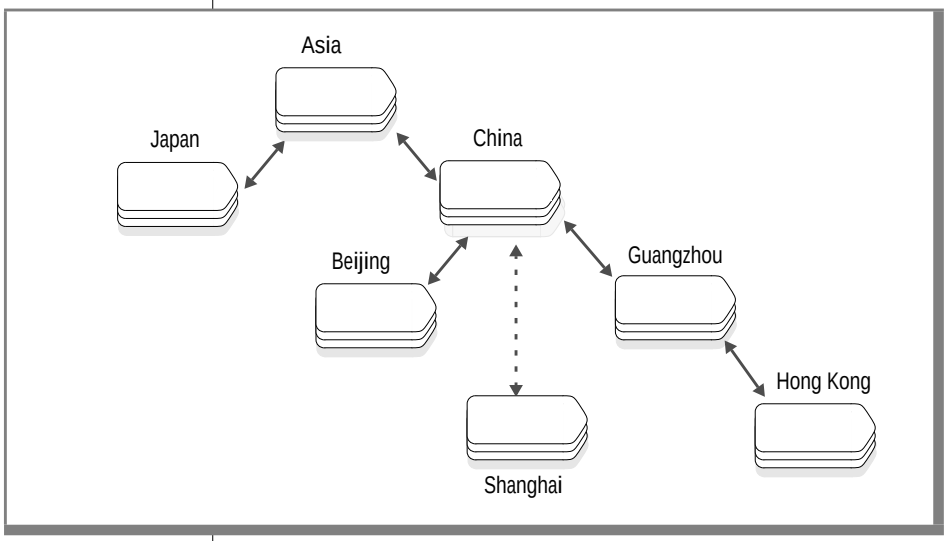



***Figure 2-3***
*Hierarchical Tree Topology*

In Figure 2-3 the parent-child relationship within the tree is as follows:

- **Asia** is the parent of **China** and **Japan**.
- **China** is the child of **Asia** *and* the parent of **Beijing**, **Shanghai**, and **Guangzhou**.
- **Guangzhou** is the child of **China** *and* the parent of **Hong Kong**.

**Asia** is the root database server. **Japan**, **China**, and **Guangzhou** are nonroot database servers. You can define **Beijing**, **Shanghai**, and **Hong Kong** as either nonroot database servers or leaf database servers, depending on how you plan to use them. The dashed connection from **China** to **Shanghai** indicates that Shanghai is a leaf server.

## Forest of Trees

A forest of trees consists of several hierarchical trees whose root database servers are fully connected. Each hierarchical tree starts with a root database server. The root database servers transfer replication messages to the other root servers for delivery to its child database servers. Figure 2-4 shows a forest of trees.

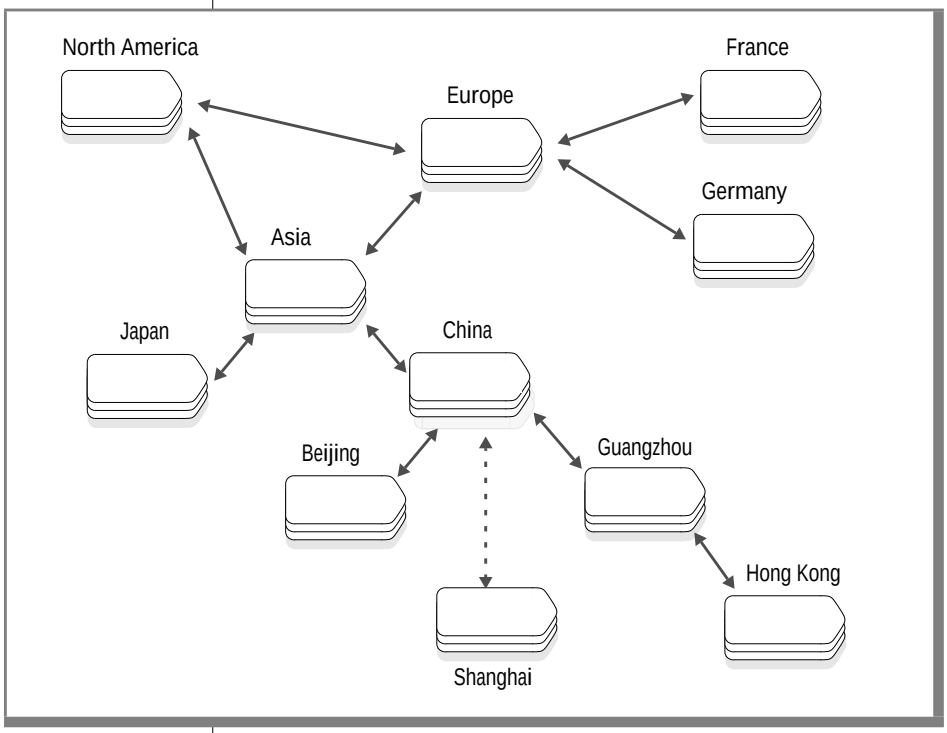



***Figure 2-4***
*Forest-of-Trees Topology*

In Figure 2-4, **North America**, **Asia**, and **Europe** are root database servers. That is, they are fully connected with each other. **France** and **Germany** are in a tree whose root is **Europe**. **Asia** is the root for the six database servers in its tree.

In a forest of trees, all replication messages from one tree to another must pass through their roots. For example, a replication message from **Beijing** to **France** must pass through **China**, **Asia**, and **Europe**.

Organizing the database servers in a hierarchical tree or a forest of trees greatly reduces the number of physical connections that are required to make a replication system. If all the database servers in Figure 2-4 were fully connected, instead of being organized in trees, 55 connections would be required.

# Enterprise Replication Objects

Enterprise Replication defines many objects that define the data in a replication system and how it is treated. The Enterprise Replication objects are:

- Enterprise Replication server
- Replicate
- Participant
- Replicate group
- Catalog

For information about defining replication objects, refer to Chapter 8, "Enterprise Replication Menus" and Chapter 11, "Command-Line Utility."

## Enterprise Replication Server

An Enterprise Replication server, or *replication server*, is an Informix database server that participates in data replication. The replication server maintains information about the replication environment, which columns should be replicated, and the conditions under which the data should be replicated. This information is stored in a database, **syscdr**, that the database server creates when it is defined as a replication server. Multiple database servers can be on the same physical computer and each database server can participate in Enterprise Replication.

## Replicate

A *replicate* defines the data (database, table, and columns) to be replicated and lists the database servers to which the data replicates. In addition to its name, a replicate contains the following information:

- Replication options
- Replicate state
- Conflict-resolution rules and scope
- Participants

### Replicate Options

When you create a replicate, you can specify the following options:

- Aborted-transaction spooling (ATS)
- Row-information spooling (RIS)
- Canonical message format
- Database triggers for replicated tables
- Replication frequency

The ATS and RIS spooling files contain information about failed transactions. You can use this information to help you diagnose problems that arise during replications. For more information about these files, refer to Chapter 10, "Diagnosing Enterprise Replication."

The canonical message format sends floating-point values in a machine-independent format when you replicate data between dissimilar hardware platforms.

For information about database triggers, refer to "Using Cascading Deletes and Triggers" on page 7-20. For information about replication frequency, refer to "Frequency Options" on page 11-14.

### Replicate State

When you define a replicate, the replicate does not begin until you explicitly change its state to *active*. For more information, refer to "Changing Replicate States" on page 8-14.

### Conflict-Resolution Rules

In update-anywhere replication, a conflict might arise because two users have updated the same table at the same time. In that case, Enterprise Replication uses the conflict-resolution rules to decide which update has precedence. You can specify any of the following conflict-resolution rules:

- Ignore
- Time stamp
- Stored procedure
- Time stamp plus stored procedure

For more information, refer to "Choosing Conflict-Resolution Rules and Scopes" on page 7-13.

### Replicate Example

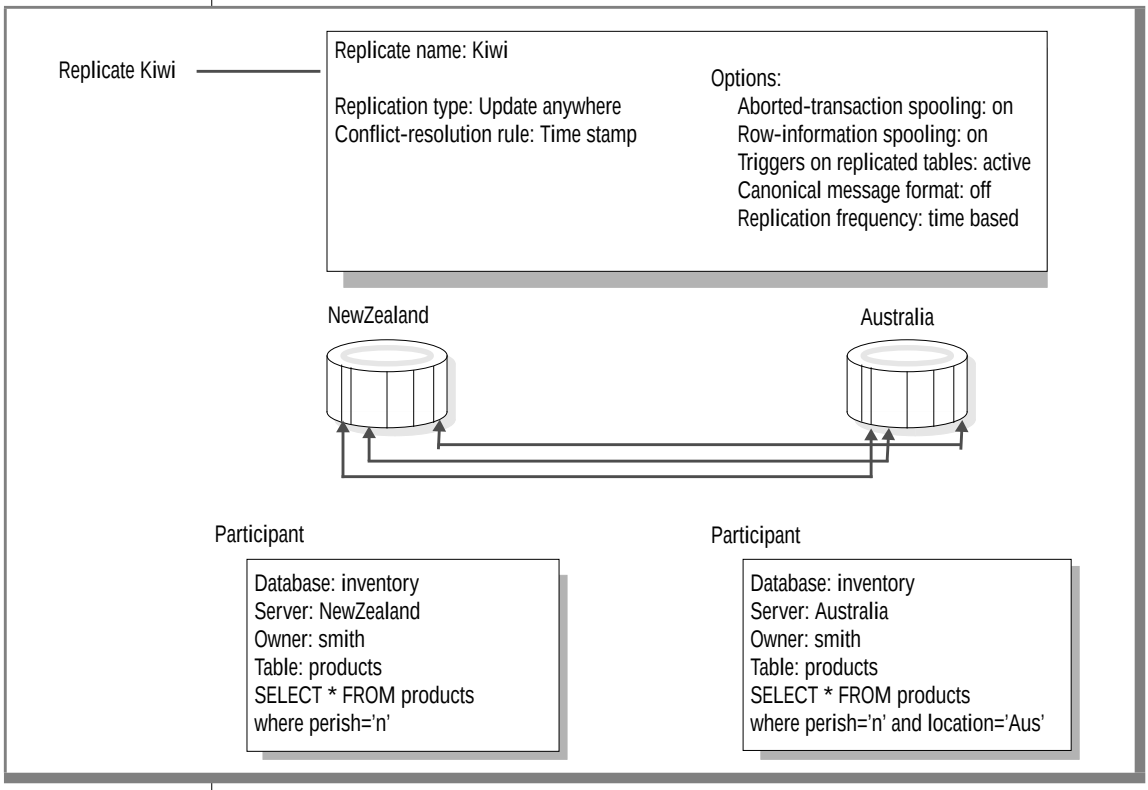
Figure 2-5 on page 2-11 shows the replicate **Kiwi**. A replicate defines the name and type of the replicate, conflict-resolution rule, and options. Each participant within the replicate identifies the data to replicate from one database server to another.

In Figure 2-5, the **inventory** database on both database servers (**NewZealand** and **Australia**) includes the table **products**, which might be defined as follows:

```
CREATE TABLE products (
    product      CHAR(20),
    prod_id      INTEGER,
.
.
    perish       CHAR(1), -- y perishable, n not-perishable
    unit_cost    DECIMAL

    )
```

Perishable items (for example, produce that has a short shelf life) are listed only in the local database. Inserts or updates on perishable items should not be replicated to other database servers.

***Figure 2-5***
*Replicate Kiwi*



## Participant

A *participant* is an entity within a replicate. A replicate contains two or more participants. The participant specifies the data that should be replicated for a single table on a single database server. A participant includes the following information:

- Database server
- Database in which the table resides
- Table name
- Table owner
- SELECT statement
- Participant type (optional)

The database server, database, table name, and table owner specify the source of the data. The SELECT statement in a participant has the following characteristics:

- The columns selected must include a primary key.
- The statement can reference only a single table.
- The statement cannot include a join or a subquery.
- The statement can include an optional WHERE clause.

The participant type can be either primary target or update anywhere. The default mode is update anywhere.

## Replicate Group

A replicate group combines several replicates that have identical participant database servers. Replicate groups provide two major benefits:

- Ease of use through easier administration
- Increased performance with parallel processing

### Ease of Use

If your replication system contains many replicates (for example, a hundred replicates) that are controlled identically, when you define a replicate group you can use a single command to start, stop, suspend, or resume all the replicates.

### Increased Performance

You can significantly increase your replication performance when you use replicate groups. Replicate groups can use parallelism to apply replicated transactions at target sites.

In a replicate group:

- each replicate must replicate to the same set of database servers.
- each replicate in the replicate group must be in the same Enterprise Replication *state*. For more information, see "Changing Replicate States" on page 8-14.

The default replication process for a replicate group is sequential processing. Nonsequential processing or parallel processing has the following two important restrictions:

- The data domains between groups must be disjoint. That is, a change to the data in one group must not affect the data in another group.
- Time dependencies between the replicates must not exist. That is, the order in which operations are performed on participants in the group must not affect the result.

***Warning:*** *If time dependencies exist or domains are not disjoint and you select non-sequential processing, you can affect data consistency and integrity.*

Before you define replicate groups, consider the following factors:

- Replicate groups have separate group-processing queues that process in parallel with all other replicate-group queues. If a transaction involves replicates that are not part of a replicate group, the replicates are processed in a single sequential queue.
- Replicate-group names are global within the replication system. Each replicate-group name must be unique. Do not define a replicate group and a replicate with the same name.

## Catalog

Each database server that participates in Enterprise Replication has a catalog in the **syscdr** database that contains information about the replication system. For all root and nonroot replication servers, this catalog is a *global* catalog that maintains a global inventory of replication system options, states, and statistical data. The global catalog contains:

- replicate definitions.
- Enterprise Replication server definitions.
- Enterprise Replication object states.
- participant definitions.
- replicate-group definitions.

The tables in one global catalog are replicated to the global catalogs of all other replication servers, thus you can manage the entire replication environment from one replication server. For information about managing replication servers (and their global catalogs), refer to Chapter 11, "Command-Line Utility" and Chapter 8, "Enterprise Replication Menus."

Leaf database servers have a limited catalog. Because the parent database server always manages operations that involve a leaf database server, the catalog of the leaf database server can contain only enough data to allow it to interact with its parent server. Limiting the catalog of leaf database servers makes the replication system more efficient because the global catalogs do not need to be replicated to the leaf servers.

# Replication Types

Enterprise Replication provides two types of data replication:

- Primary target
- Update anywhere

For a discussion of ways in which these replication types are used, refer to Chapter 4, "Selecting Enterprise-Replication Systems." For examples of simple replications, refer to Chapter 3, "Replication Examples."

## Primary-Target Replication

In primary-target replication, all database changes originate at the primary database and are replicated to the target databases. Changes at the target databases are not replicated to the primary. A primary-target replication system can provide one-to-many or many-to-one replication.

### *One-to-Many Replication*

In the one-to-many, or *distribution*, replication, all changes to a primary database server are replicated to many target database servers. You might use this replication model when information gathered at a central site must be disseminated to many scattered sites.

### *Many-to-One Replication*

In many-to-one replication, or *consolidation* replication, many primary servers send information to a single target server. You might use this replication model when many sites are gathering information (for example, local field studies for an environmental study) that needs to be centralized for final processing.

## Update-Anywhere Replication

In update-anywhere replication, changes made on any participating database server are replicated to all other participating database servers. When you use update-anywhere replication, users at two or more different database servers might update the same row at the same time. To cope with this situation, you must provide rules for *conflict resolution*.

# Conflict Resolution

When multiple database servers try to update the same row at the same time, Enterprise Replication must determine which new data to replicate. To solve conflict resolution, you must specify a conflict-resolution *rule* and the conflict-resolution *scope*.

For a detailed discussion of conflict resolution, refer to "Evaluating Data for Replication" on page 7-4.

## Conflict-Resolution Rules

A conflict occurs when two database servers attempt to modify the same piece of data. Enterprise Replication has the following four conflict-resolution rules:

- **Ignore**. The ignore rule does not attempt to detect conflicts.
- **Time stamp**. The row or transaction with the most recent time stamp is applied.
- **Stored procedure**. An SPL procedure that you provide determines which data should be applied.

  Procedures for conflict-resolution must be in SPL. Enterprise Replication does not allow user-defined routines in C or in Java.
- **Time stamp with stored procedure**. If the time stamps are identical, Enterprise Replication uses an SPL procedure to resolve the conflict.

## Resolution Scope

Each conflict-resolution rule behaves differently depending on the conflict-resolution *scope* you chose. The following list describes each conflict-resolution scope:

- **Row-by-row scope**. Enterprise Replication resolves conflicts one row at a time.
- **Transaction scope**. Enterprise Replication applies conflict rules to the entire transaction. *Transaction scope for conflict resolution guarantees transactional integrity.*

# Replication Data

Enterprise Replication uses a log-based transaction-capture mechanism to gather data for replication. A log-based transaction capture minimizes the effect on transaction performance; it captures changes from the logical log instead of competing with transactions that access production tables.

Enterprise Replication reads the logical log to obtain the row images for tables that participate in a replicate. These rows are passed to Enterprise Replication for further evaluation.

## Evaluating Data for Replication

Enterprise Replication evaluates transactions based on a distinct time and a change to a final image of a row.

### *Evaluating Time*

All Enterprise Replication commands are issued at distinct times. Enterprise Replication records the time the command is issued and uses that time when it determines which transactions are eligible for replication.

### *Evaluating Images of a Row*

After Enterprise Replication evaluates the recorded time to determine whether a transaction should be replicated, Enterprise Replication evaluates the initial and final images of a row and any changes that occur between the two row images to determine which rows to replicate.

## Processing in Parallel

Enterprise Replication evaluates the images of rows in parallel to assure high performance. The records that qualify are collected into a transaction list and then passed to the Enterprise Replication reliable-message queue for distribution. Figure 2-6 illustrates how Enterprise Replication uses parallel processing to evaluate transactions for replication.

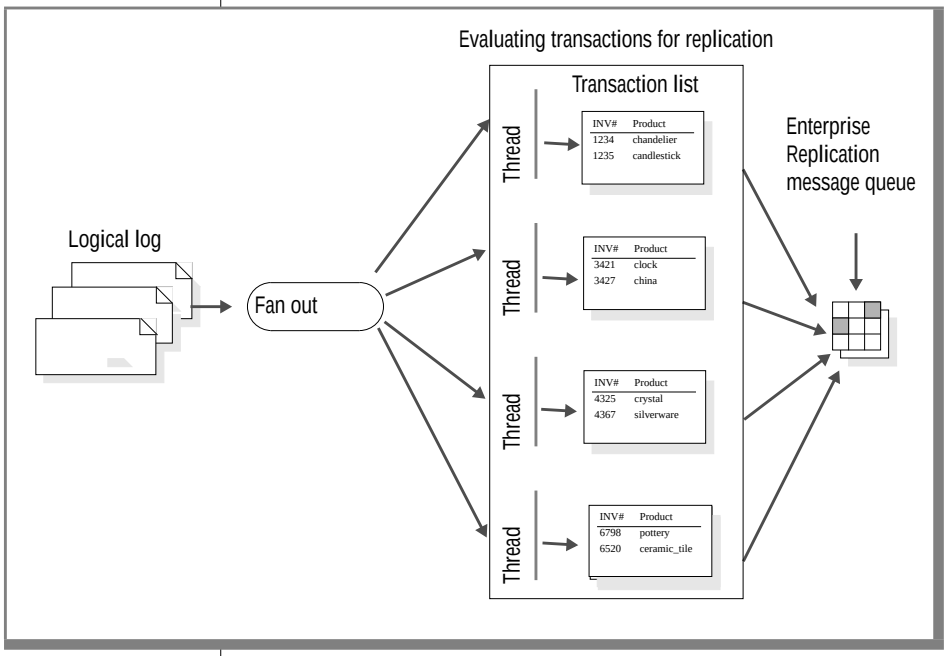



***Figure 2-6*** *Processing in Parallel for High Performance*

After Enterprise Replication identifies transactions that qualify for replication, Enterprise Replication transfers the transaction data to a message queue.

## Distributing Data

Enterprise Replication uses the message queue to ensure that all data reaches the appropriate server, regardless of a network or system failure. In the event of a failure, the message queue stores data until the network or system is operational. The message queue replicates data efficiently with a minimum of data copying and sending.

## Applying Replicated Data

Enterprise Replication uses a data-synchronization process to apply the replicated data to target database servers. The target database servers acknowledge receipt of a message when the message is *safely stored*. A message is safely stored when it is in a stable (recoverable) queue or when it is in the target database. The data-synchronization process ensures that transactions are applied at the target database servers in the same order as they were committed on the source database server.

Chapter

# 3

# Replication Examples


In This Chapter . . . . . 3-3

Environment for the Replication Examples . . . . . 3-3

Terminology . . . . . 3-5

Primary-Target Example Using the CLU . . . . . 3-6
- Preparing for a Primary-Target Replication . . . . . 3-6
- Cause a Replication . . . . . 3-8
- Do Not Cause a Replication . . . . . 3-9

Update-Anywhere Example Using the CLU . . . . . 3-9
- Preparing for an Update-Anywhere Replication . . . . . 3-9
- Cause a Replication . . . . . 3-12
  - Changing the stock Table on the usa Database Server . . . . 3-12
  - Updating the stock Table on the japan Database Server. . . . 3-13

Hierarchy Example Using the CLU . . . . . 3-13
- Preparing for a Hierarchy Example . . . . . 3-14
- Defining a Hierarchy . . . . . 3-14

Primary-Target Example Using ERM. . . . . 3-16
- Removing Previously Defined Replication Servers . . . . . 3-17
  - Deleting Previously Defined Replication Servers . . . . . 3-17
  - Using Previously Defined Replication Servers . . . . . 3-18
- Defining the First Replication Server . . . . . 3-18
  - The Declare Server Dialog Box for NonExpert Mode . . . . 3-20
  - The Declare Server Dialog Box for Expert Mode . . . . . 3-22
  - Choosing Nonexpert Mode . . . . . 3-23
  - Displaying the List of Servers . . . . . 3-23
- Defining a Replication Server on italy . . . . . 3-24
- Creating the Primary-Target Replicate . . . . . 3-26

Choosing a Name for the Replicate . . . . . 3-26
Specifying the Replication Frequency . . . . . 3-26
Selecting the Participants . . . . . 3-28
Selecting the Primary . . . . . 3-29
Selecting a Table . . . . . 3-30
Selecting the Columns . . . . . 3-31
Summary . . . . . 3-32
Viewing the Replicate. . . . . 3-33
Activating the Replicate . . . . . 3-34
Cause a Replication . . . . . 3-36
Do Not Cause a Replication . . . . . 3-36

Update-Anywhere Example Using ERM . . . . . 3-37
Changing to Expert Mode . . . . . 3-37
Creating an Update-Anywhere Replicate . . . . . 3-37
Choosing a Replicate Name and Type . . . . . 3-37
Specifying a Conflict Rule . . . . . 3-38
Specifying the Replication Options . . . . . 3-39
Specifying the Frequency . . . . . 3-39
Selecting the Participants . . . . . 3-40
Participants Identical . . . . . 3-41
Selecting a Table . . . . . 3-41
Selecting the Columns . . . . . 3-42
Summary . . . . . 3-43
Viewing the Replicate. . . . . 3-44
Defining a Replication Server on japan . . . . . 3-45
Adding a Participant to repl4 . . . . . 3-46
Selecting a New Participant . . . . . 3-47
Specifying a Table and Columns. . . . . 3-48
Activating the Replicate . . . . . 3-48
Cause a Replication . . . . . 3-49

Hierarchy Example Using ERM . . . . . 3-49
Deleting the Previously-Defined Servers . . . . . 3-49
Defining the Nonroot Replication Servers . . . . . 3-49
Defining the boston Nonroot Server . . . . . 3-50
Defining the denver Nonroot Server . . . . . 3-50
Defining a Leaf Replication Server . . . . . 3-50
Displaying the Hierarchy . . . . . 3-50
Viewing the Server Properties . . . . . 3-51

## In This Chapter

This chapter contains simple examples of replication. The first three examples use the *command-line utility* (CLU) that allows you to control Enterprise Replication from the command line. Chapter 11, "Command-Line Utility," documents the CLU.

The second three examples use the Enterprise Replication Manager (ERM), a graphical user interface (GUI) that runs on Windows NT, to perform the same replications as the first three examples. Chapter 8, "Enterprise Replication Menus," and Chapter 9, "Enterprise Replication Manager Wizards," document ERM.

## Environment for the Replication Examples

To run the replication examples in this chapter, you need three Informix database servers. Each database server must be in a database server group.

This example uses the following environment:

- Three UNIX computers (**s1**, **s2**, and **s3**) are hosts for the database servers **usa**, **italy**, and **japan**, respectively. Each computer has active network connections to the other two computers.
- The database servers **usa**, **italy**, and **japan** are members of the database server groups **g_usa**, **g_italy**, and **g_japan**, respectively.

  For information about database server groups, see "Database Server Groups" on page 12-3.

UNIX

- The UNIX **sqlhosts** file for each database server contains the following connectivity information:

```
g_usa     group      -    -            i=1
usa       ontlitcp   s1   techpubs1    g=g_usa
g_italy   group      -    -            i=8
italy     ontlitcp   s2   techpubs8    g=g_italy
g_japan   group      -    -            i=6
japan     ontlitcp   s3   techpubs6    g=g_usa
```

♦

WIN NT

With Dynamic Server, Version 7.31, use the IECC Add Database Servers wizard to prepare the SQLHOSTS connectivity information.

With Dynamic Server, Version 9.2, refer to the documentation notes described in "Documentation Notes, Release Notes, Machine Notes" on page 15 of the Introduction for information about preparing the SQLHOSTS connectivity information. ♦

- All commands in this example, except for creation of the sample databases on **italy** and **japan**, are issued from the computer **s1**.
- The databases for the examples in this chapter are identical **stores_demo** databases with logging, as follows:

  Create a database named **stores** on the **usa** database server:

```
s1> dbaccessdemo -log stores
```

  Create a database named **stores** on the **italy** database server:

```
s1> rlogin s2
s2> dbaccessdemo -log stores
```

  Create a database named **stores** on the **japan** database server:

```
s1> rlogin s3
s2> dbaccessdemo -log stores
```

# Terminology

Enterprise Replication and ERM deal with the following three distinct objects that are treated almost as synonyms:

- Database servers
- Database server groups
- Replication servers

A replication server is a database server that includes a database that describes the replication environment. Thus, every replication server is a database server, but not every database server is a replication server.

A database server group includes one or more database servers. For some purposes, all database servers in a database server group are treated as the same server. In Enterprise Replication, a database server group includes more than one database server *only* if a computer has multiple network interface cards. However, a database server group can include multiple database server aliases. If you use DBSERVERNAME and DBSERVERALIASES, the DBSERVERNAME should correspond to the SQLHOSTS entry that specifies the network connection. That is, DBSERVERNAME should not correspond to a shared-memory connection. For information about database server aliases, refer to your *Administrator's Guide*.

This manual shows database servers and database server groups explicitly. A database server name, such as **myserver**, never has a prefix. In this manual the database server group to which **myserver** belongs is *always* prefixed with **g_**: for example, **g_myserver**. However, ERM usually asks you to enter the name of, or select, a *server*. In most cases ERM uses *server* to mean database server group.

To decide what name to use, use the following guidelines:

- To describe the contents of a table that should be replicated, use the name of the database server.
- In all other cases, use the name of the database server group.

# Primary-Target Example Using the CLU

This section contains a simple example of *primary-target* replication. In primary-target replication, only changes to the primary table are replicated to the other tables in the replicate. Changes to the secondary table(s) are not replicated.

## Preparing for a Primary-Target Replication

The following example defines the **g_usa** and **g_italy** database server groups as Enterprise Replication servers and creates a replicate, **repl1**:

```
1  cdr define server --init g_usa
2  cdr list server
3  cdr define server --connect=italy --init --sync=g_usa g_italy
4  cdr list server
5  cdr define replicate --conflict=ignore repl1 \
6  "P stores@g_usa:informix.manufact" "select * from manufact" \
7  "R stores@g_italy:informix.manufact" "select * from manufact"
8  cdr list replicate
9  cdr start replicate repl1
10 cdr list replicate
```

### *Line 1*

Before replicating data, you must define the database servers as *replication servers*. A replication server is a database server that has an extra database that holds replication information. Line 1 creates and populates the database that defines the **usa** database server as a replication server. The **--init** option specifies that this server is a new replication server. When you define a replication server, you *must* use the name of the database server group (**g_usa**) rather than the database server name.

### *Line 2*

This line displays the replication server that you defined, so that you can verify that the definition succeeded. The command returns the following information:

```
SERVER    ID STATE    STATUS    CONNECTION CHANGED
------------------------------------------------------------
g_usa      1 Active   Local
```

### Line 3

Define the second database server, **italy**, as a replication server. The **--connect** option allows you to define **italy** (on the **s2** computer) while working at the **s1** (**usa**) computer. The **--sync** option instructs the command to use the already-defined replication server (**g_usa**) as a pattern for the new definition. The **--sync** option also links the two replication servers into a replication environment.

***Tip:*** *In all options except the* ***--connect*** *option, Enterprise Replication uses the name of the database server group to which a database server belongs instead of the name of the database server itself.*

### Line 4

Verify that the second definition succeeded. The command returns the following information:

```
SERVER    ID STATE    STATUS    CONNECTION CHANGED
------------------------------------------------------------
g_italy    8 Active    Connected  JUN 14 14:38:44 1999
g_usa      1 Active    Local
```

### Lines 5 through 7

Lines 5 through 7 define a replicate. These lines are all one command. The backslashes (\) at the end of lines 5 and 6 indicate that the command continues on the next line. Line 5 names the replicate, **repl1**, and specifies that conflicts should be ignored.

Line 6 describes one participant in the replicate. The `P` indicates that in this replicate **usa** is a primary server. That is, if any data in the selected columns changes, that changed data should be sent to the secondary server(s).

Line 7 describes a second participant in the replicate. The `R` indicates that in this replicate **italy** is a secondary server. The specified table and columns receive information that is sent from the primary server. Changes to those columns on **italy** are *not* replicated.

### Line 8

This line displays the replicate that you defined, so that you can verify that the definition succeeded. The command returns the following information:

```
REPLICATE     STATE       CONFLICT     FREQUENCY    OPTIONS
-------------------------------------------------------------
repl1         INACTIVE    ignore       immediate    row
```

### Line 9

The display for line 8 shows that the STATE of the replicate is INACTIVE. Lines 5 through 7 *describe* a replicate but do not make the replicate active. Line 9 makes the replicate active. If any changes are made to the **manufact** table, the changes will be replicated to the other participants in the replicate.

### Line 10

Line 10 displays the replicate so that you can verify that the STATE has changed to ACTIVE. The command returns the following information:

```
REPLICATE     STATE       CONFLICT     FREQUENCY    OPTIONS
-------------------------------------------------------------
repl1         ACTIVE    ignore       immediate    row
```

## Cause a Replication

Now you can modify the **manufact** table on the **usa** database server and see the change reflected in the **manufact** table on the **italy** database server.

Use DB-Access to insert a value into the **manufact** table on **usa**:

```
INSERT INTO stores@usa:manufact VALUES ('AWN','Allwyn','8');
```

Now observe the changes on **usa** and on **italy**:

```
SELECT * from stores@usa:manufact
SELECT * from stores@italy:manufact
```

## Do Not Cause a Replication

In **repl1**, **usa** is the primary database server and **italy** is the target. Changes made to the **manufact** table on **italy** do not replicate to **usa**.

Use DB-Access to insert a value into the **manufact** table on **italy**:

```
INSERT INTO stores@italy:manufact VALUES ('ZZZ','Zip','9');
```

Now verify that the change occurred on **italy** but did not replicate to the **manufact** table on **usa**:

```
SELECT * from stores@usa:manufact
SELECT * from stores@italy:manufact
```

# Update-Anywhere Example Using the CLU

This example creates a simple *update-anywhere* replication. In update-anywhere replication, changes to *any* table in the replicate are replicated to all other tables in the replicate. In this example, any change to the **stock** table of the **stores** database on any database server in the replicate will be replicated to the **stock** table on the other database servers.

## Preparing for an Update-Anywhere Replication

```
 1 cdr define replicate --conflict=ignore repl2 \
 2 "stores@g_usa:informix.stock" "select * from stock" \
 3 "stores@g_italy:informix.stock" "select * from stock"
 4 cdr list replicate
 5 cdr list replicate repl2
 6 cdr define server --connect=japan --init --sync=g_usa g_japan
 7 cdr list server
 8 cdr change replicate --add repl2 \
 9 "stores@g_japan:informix.stock" "select * from stock"
10 cdr list replicate repl2
11 cdr start replicate repl2
12 cdr list replicate
```

### Lines 1 through 3

Lines 1 through 3 define a replicate. These lines are all one command. The backslashes (\) at the end of lines 1 and 2 indicate that the command continues on the next line.

Line 1 names the replicate, **repl2**, and specifies that conflicts should be ignored.

Line 2 describes one *participant* in the replicate. It specifies the table and the columns to replicate on database server **usa**. Line 3 describes a second participant in the replicate. It specifies the table and the columns to replicate on database server **italy**.

If any data in the selected columns changes, on either participant, that changed data should be sent to the other participants in the replicate.

The replicate definition could also include **japan**, but this example adds **japan** in a separate step on lines 8 and 9.

### Line 4

This line displays all the replicates so that you can verify that your definition of **repl2** succeeded. The command returns the following information:

```
REPLICATE     STATE       CONFLICT     FREQUENCY    OPTIONS
------------------------------------------------------------
repl1         ACTIVE      ignore       immediate    row
repl2         INACTIVE    ignore       immediate    row
```

### Line 5

Although line 4 shows that **repl2** exists, it does not give much information about the replicate itself. Line 5 shows the participants (**usa** and **italy**) and the *participant modifiers* (the SELECT statements) for **repl2**. The command returns the following information:

```
REPLICATE SERVER  STATE    TABLE SELECT
------------------------------------------------------------
repl2   g_usa    INACTIVE  manufact  select * from stock
repl2   g_italy  INACTIVE  manufact  select * from stock
```

### *Line 6*

Line 6 adds the **japan** database server to the replication system already defined in the previous example. You can use either **g_usa** or **g_italy** in the **--sync** option.

Enterprise Replication maintains identical information on all servers that participate in the replication system. Therefore, when you add the **japan** database server, information about that server is propagated to all previously-defined replication servers (**usa** and **italy**).

### *Line 7*

This line displays the replication servers so that you can verify that the definition succeeded. The command returns the following information:

```
SERVER    ID STATE    STATUS    CONNECTION CHANGED
------------------------------------------------------------
g_italy    8 Active    Connected  JUN 14 14:38:44 1999
g_japan    6 Active    Connected  JUN 15 9:12:27 1999
g_usa      1 Active    Local
```

### *Lines 8 and 9*

Lines 8 and 9 together form one command. Line 8 specifies an addition to **repl2**. Line 9 gives the participant and participant modifier that should be added.

### *Line 10*

Line 10 gives detailed information about **repl2**. After the addition of the participant in lines 8 and 9, the command returns the following information:

```
REPLICATE SERVER  STATE    TABLE SELECT
------------------------------------------------------------
repl2  g_usa    INACTIVE  manufact  select * from stock
repl2  g_italy  INACTIVE  manufact  select * from stock
repl2  g_japan  INACTIVE  manufact  select * from stock
```

### Line 11

The display for line 10 shows that the STATE of **repl2** is INACTIVE. Line 11 makes the replicate active. Changes made to the **stock** table on **usa** are reflected in the **stock** tables on both **italy** and **japan**.

### Line 12

Line 12 displays a list of replicates so that you can verify that the STATE of **repl2** has changed to ACTIVE. The command returns the following information:

```
REPLICATE     STATE     CONFLICT     FREQUENCY    OPTIONS
------------------------------------------------------------
repl1         ACTIVE    ignore       immediate    row
repl2         ACTIVE    ignore       immediate    row
```

## Cause a Replication

Now you can modify the **stock** table on one database server and see the change reflected on the other database servers.

### Changing the stock Table on the usa Database Server

Use DB-Access to insert a line into the **stock** table on **usa**:

```
INSERT INTO stores@usa:stock VALUES (401, “PRC”, “ski boots”,
200.00, “pair”, “pair”);
```

Observe the change on the **italy** and **japan** database servers:

```
SELECT * from stores@italy:stock;
SELECT * from stores@japan:stock;
```

### Updating the stock Table on the japan Database Server

Use DB-Access to change a value in the **stock** table on **japan**:

```
UPDATE stores@japan:stock SET unit_price = 190.00
WHERE stock_num = 401;
```

Verify that the change has replicated to the **stock** table on **usa** and on **italy**:

```
SELECT * from stores@usa:stock WHERE stock_num = 401;
SELECT * from stores@italy:stock WHERE stock_num = 401;
```

## Hierarchy Example Using the CLU

The example in this section adds a replication tree to the fully-connected environment of the **usa**, **italy**, and **japan** replication servers. The nonroot servers **boston** and **denver** are children of **usa**. (The leaf server **miami** is a child of **boston**.) Figure 3-1 shows the replication hierarchy.

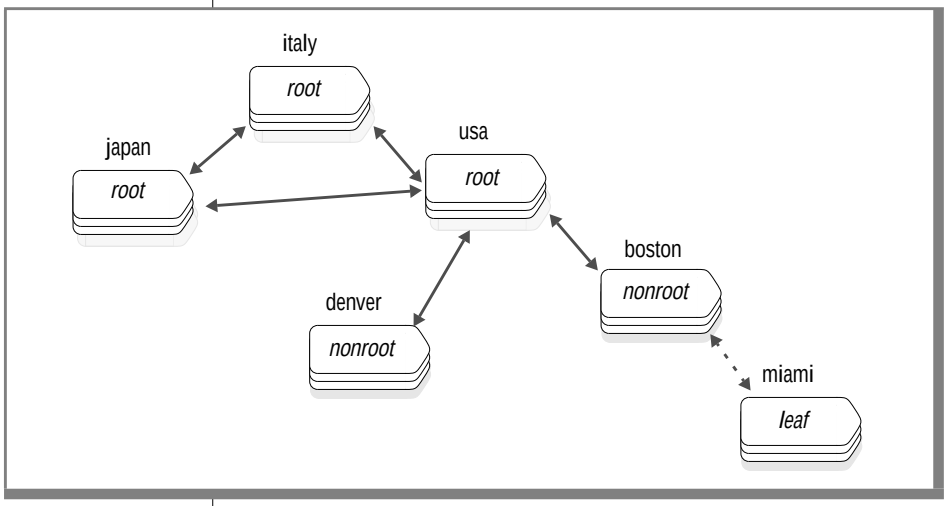



***Figure 3-1***
*Hierarchical Tree Example*

## Preparing for a Hierarchy Example

To try this example, you need to prepare three additional database servers: **boston**, **denver**, and **miami**. To prepare the database servers, use the techniques described in "Environment for the Replication Examples" on page 3-3.

## Defining a Hierarchy

The following example defines a replication hierarchy that includes **denver**, **boston**, and **miami** and whose root is **usa**.

```
1  cdr define server --connect boston --nonroot --init \
2  --sync g_usa g_boston
3  cdr define server -c denver -I -N --ats=/ix/myats \
4  -S g_usa g_denver
5  cdr list server
6  cdr list server --connect denver
7  cdr define server -c miami -I --leaf -S g_boston g_miami
8  cdr list server -c miami
9  cdr list server g_usa
```

### *Lines 1 and 2*

Lines 1 and 2 add **boston** to the replication hierarchy as a nonroot server attached to the root server **usa**. The backslash (\) at the end of line 1 indicates that the command continues on the next line.

### *Lines 3 and 4*

Lines 3 and 4 add **denver** to the replication hierarchy as a nonroot server attached to the root server **usa**. This command uses short forms for the **connect**, **init**, and **sync** options. (For information about the short forms, refer to "Option Abbreviations" on page 11-6.) The command also specifies a directory for collecting information about failed replication transactions, **/ix/myats**.

### *Line 5*

Line 5 lists the replication servers as seen by the **usa** replication server. The root server **usa** is fully connected to all the other root servers. Therefore **usa** knows the connection status of all other root servers and of its two child servers, **denver** and **boston**. The command returns the following information:

```
SERVER       ID STATE    STATUS     CONNECTION CHANGED
-------------------------------------------------------
g_boston      3 Active   Connected  Aug19 14:20:03 1999
g_denver     27 Active   Connected  Aug19 14:20:03 1999
g_italy       8 Active   Connected  Aug19 14:20:03 1999
g_japan       6 Active   Connected  Aug19 14:20:03 1999
g_usa         1 Active   Local
```

### *Line 6*

Line 6 lists the replication servers as seen by the **denver** replication server. The nonroot server **denver** has a complete global catalog of replication information, so it knows all the other servers in its replication system. However, **denver** knows the *connection status* only of itself and its parent, **usa**. The command returns the following information:

```
SERVER       ID STATE    STATUS     CONNECTION CHANGED
-------------------------------------------------------
g_boston      3 Active
g_denver     27 Active   Local
g_italy       8 Active
g_japan       6 Active
g_usa         1 Active   Connected  Aug 19 14:28:22 1999
```

### *Line 7*

Line 7 defines **miami** as a leaf server whose parent is **boston**.

### Line 8

Line 8 lists the replication servers as seen by **miami**, a leaf server. As a leaf replication server, **miami** has a limited catalog of replication information. It knows only about itself and its parent. The command returns the following information:

```
SERVER        ID STATE    STATUS     CONNECTION CHANGED
------------------------------------------------------
g_boston       3 Active   Connected  Aug19 14:35:17 1999
g_miami        4 Active   Local
```

### Lines 9

Line 9 lists details about the **usa** replication server. The server is a *hub*; that is, it forwards replication information to and from other servers. It uses the default values for time-out, sendq, recvq, and atsdir. The command returns the following information:

```
NAME      ID  ATTRIBUTES
-------------------------------------
g_usa      1  timeout=15 hub sendq=rootdbs recvq=rootdbs
atsdir=/tmp
```

WIN NT

## Primary-Target Example Using ERM

This example uses the Enterprise Replication Manager (ERM) to perform the same simple *primary-target* replication that the "Primary-Target Example Using the CLU" on page 3-6 shows.

To use ERM with Dynamic Server, Version 7.31, install Informix Enterprise Command Center (IECC). From IECC, choose **Tools→Enterprise Replication Manager**. To add or modify the SQLHOSTS connectivity information, use the IECC Add Database Server wizard.

To use ERM with Dynamic Server, Version 9.2, install ERM and the proper connectivity tools. For information about preparing ERM for Version 9.2, refer to the documentation notes described in "Documentation Notes, Release Notes, Machine Notes" on page 15 of this chapter.

## Removing Previously Defined Replication Servers

After you define a database server as a replication server, it remains a replication server. Therefore, if you have already done the examples in the first part of this chapter and want to start this example from the beginning, you must remove the replication servers that you defined in the previous examples.

If you do not want to start from the beginning, skip the next topic and go to "Using Previously Defined Replication Servers" on page 3-18 and continue with the rest of the example.

### *Deleting Previously Defined Replication Servers*

You can remove the replication servers in either of the following two ways:

- If you are using *practice* database servers, completely re-initialize each database server, as follows:
  - Use **oninit -iy** to reinitialize the database and erase all previous data.
  - Re-create the **stores** databases, as shown in the final bullet of "Environment for the Replication Examples" on page 3-3.
- If you cannot re-initialize the servers, delete each replication server. You can delete all the servers from the **s1** computer (the host for **g_usa**).

  For each server except **usa**, issue the following commands:

  ```
  cdr delete server server_group_name
  cdr delete server -c server_name server_group_name
  ```

  For example:

  ```
  cdr delete server g_japan
  cdr delete server -c japan g_japan
  ```

  Finally, delete the replication server on **usa**:

  ```
  cdr delete server g_usa
  ```

The preceding commands remove *only* the definition of the replication servers. They do *not* halt the database server, remove any other databases, or negate the changes that you made to the **stores** databases in the earlier examples.

After you remove the definitions of the replication servers, continue with "Defining the First Replication Server" on page 3-18.

### Using Previously Defined Replication Servers

The examples in "Preparing for a Primary-Target Replication" on page 3-6 and "Preparing for an Update-Anywhere Replication" on page 3-9 defined the database servers **usa**, **italy**, and **japan** as replication servers. You can use these replication servers if you do not want to redefine the replication servers with ERM.

**To continue without redefining the replication servers**

1. Start ERM. For information about starting ERM, refer to "Primary-Target Example Using ERM" on page 3-16.
2. Select the **g_usa** replication server from the server dialog box.
3. Skip to "Creating the Primary-Target Replicate" on page 3-26.

***Tip:*** *If you continue without redefining the replication servers, the displays in ERM will include* ***repl1*** *and* ***repl2****.*

## Defining the First Replication Server

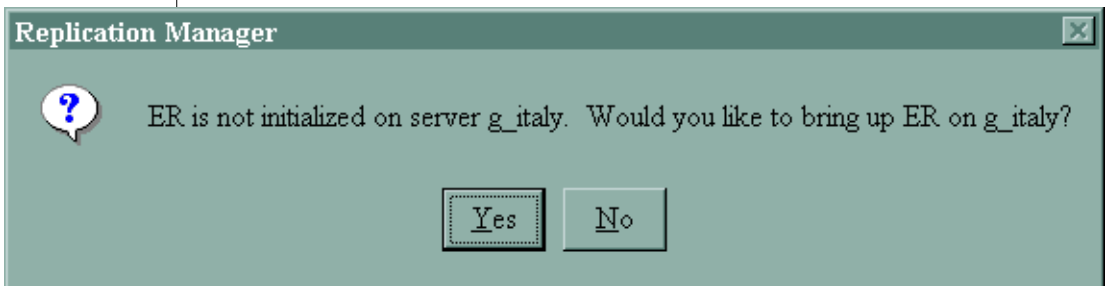
Before you replicate data, you must define the database servers as *replication servers*. A replication server is a database server that has an extra database, **syscdr**, that holds replication information.

***Important:*** *This section assumes that you have not yet defined any replication servers.*

When you open the Replication Manager, it checks to see whether you defined any database servers for Enterprise Replication. If you defined at least one replication server, the Replication Manager main window appears.

If you did not define any replication servers (that is, if you are starting replication for the first time), you must define a database server for replication before you can continue. ERM displays the dialog box that Figure 3-2 shows.

***Figure 3-2***
*Server Not Initialized Dialog Box*

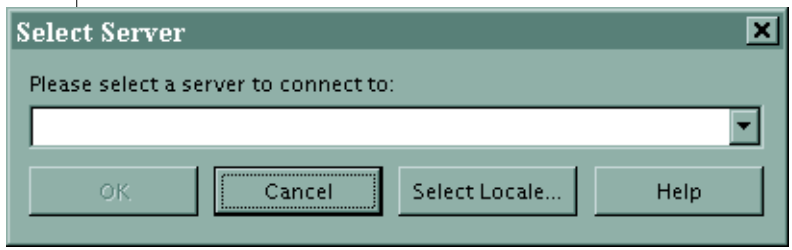



At this point, you could click **Yes** and define **g_italy**. However, to parallel the example in "Update-Anywhere Example Using the CLU" on page 3-9, **g_usa** should be defined first. Click **No**.

***Tip:*** *ERM might display one of the other database server groups. Use the Select Server dialog box to select* ***g_usa****.*

ERM displays the Select Server dialog box that Figure 3-3 shows.

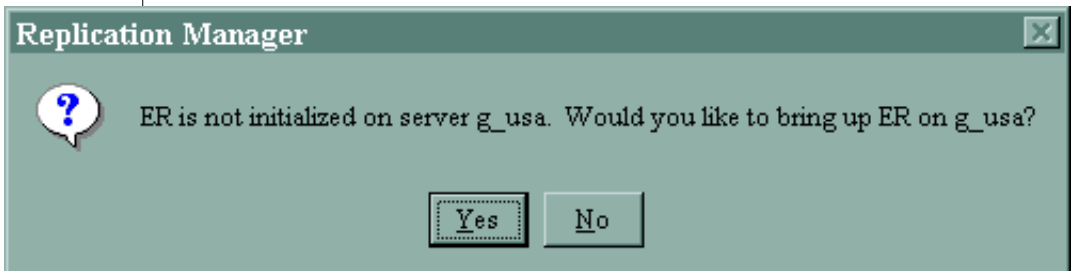
***Figure 3-3***
*Select Server Dialog Box*




1. Click the down arrow to display the list of server groups.
2. Select **g_usa**.
3. Click **OK**.

ERM displays the dialog box that Figure 3-4 shows. Click **Yes** to define the replication server **g_usa**.

***Figure 3-4***
*Server Not Initialized Dialog Box*




### *The Declare Server Dialog Box for NonExpert Mode*

ERM displays the Declare Server dialog box. ERM has two modes, expert mode and nonexpert mode. When you first install ERM, it is in nonexpert mode.

If you are in nonexpert mode, Figure 3-5 shows the Declare Server dialog box.

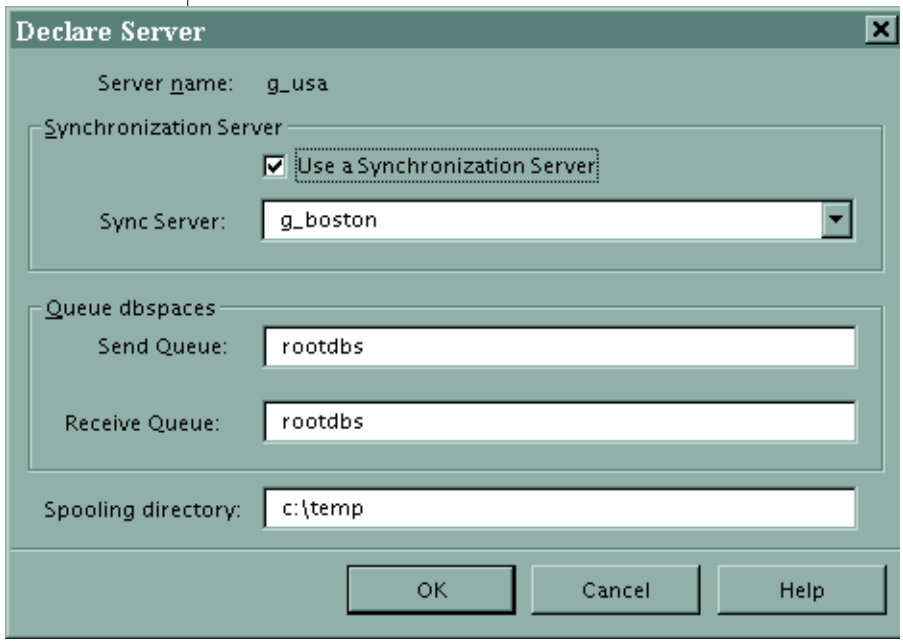



***Figure 3-5***
*Declare Server Dialog Box for Nonexpert Mode*

To declare the **usa** replication server:

- Remove the check mark from Use a Synchronization Server.
- Change the entry in the Spooling directory box to **/tmp**. (Remember, the **usa** database server is on a UNIX computer.)
- Leave the other values set to the defaults.

  For information about the Queue dbspaces, refer to "Receiving and Sending Data from the Message Queue" on page 7-11.
- Click **OK** to continue.

### The Declare Server Dialog Box for Expert Mode

If you are in expert mode, follow these steps to declare the **usa** replication server:

- Leave the default choice, Root Server, in the Server Type box.
- Remove the check mark from Use a Synchronization Server in the Synchronization Server box.
- Leave the other values set to the defaults.

  When you finish, the dialog box should look like Figure 3-6.
- Click **OK** to continue.

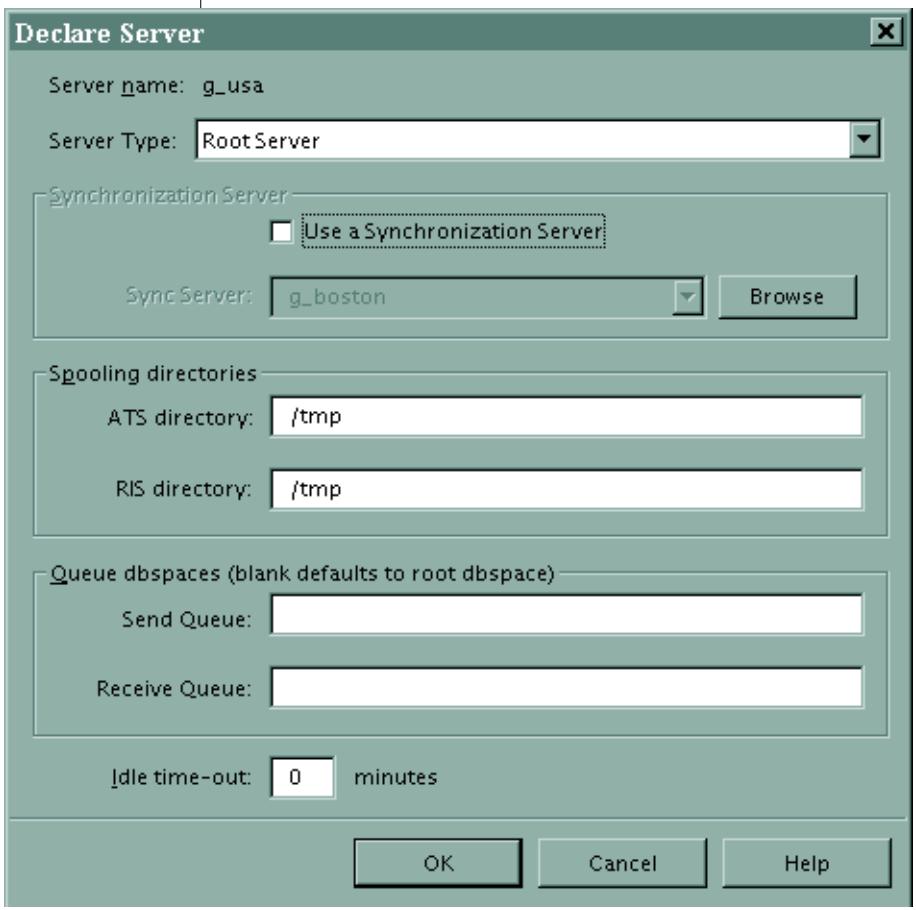



***Figure 3-6***
*Declare Server Dialog Box for Expert Mode*

### Choosing Nonexpert Mode

The rest of this example assumes that you are in nonexpert mode. If you are in expert mode, the instructions and the figures will not match your display.

When the main ERM window reappears, select **Window→Expert Mode** to remove the check beside Expert. If there is no check, as in Figure 3-7, you are already in nonexpert mode.

***Figure 3-7***
*Window Menu*

### Displaying the List of Servers

To display the list of servers, click the database server icon, , in the upper-left part of the ERM display. Figure 3-8 shows the replication servers. At this point, the only replication server is **g_usa**.

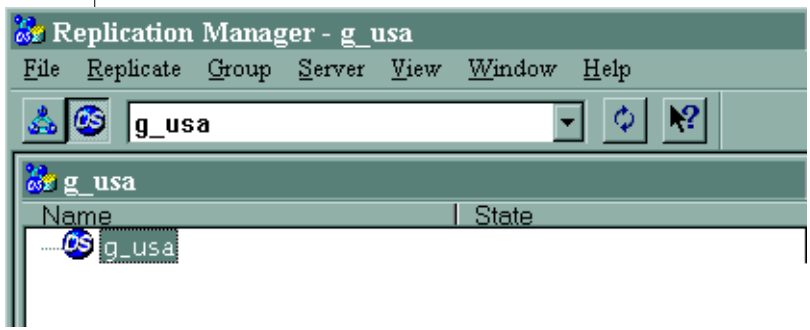



***Figure 3-8***
*List of Replication Servers*

## Defining a Replication Server on italy

To define the **italy** database server as a replication server, select **g_italy** from the Server list box, as Figure 3-9 shows.

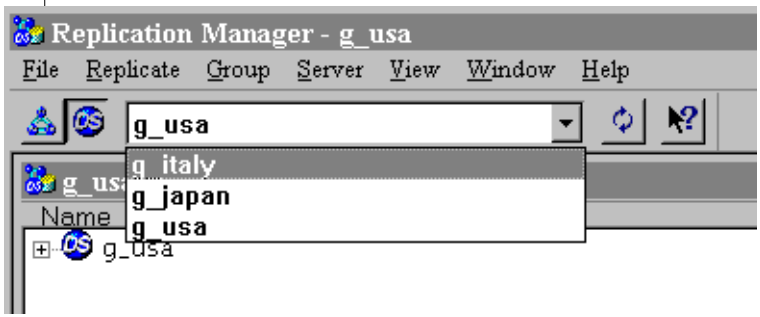



***Figure 3-9***
*Select Server List Box*

The **italy** database server is not yet a replication server. ERM asks if you would like to bring up Enterprise Replication on **g_italy**. Click **Yes**.

The second and subsequent replication servers must be linked to the first replication server (**g_usa**) so they can exchange information. To create the connection between the replication servers, choose **g_usa** as the synchronization server. Set the spooling directory to **/tmp**. Figure 3-10 shows the finished dialog box. Click **OK**.

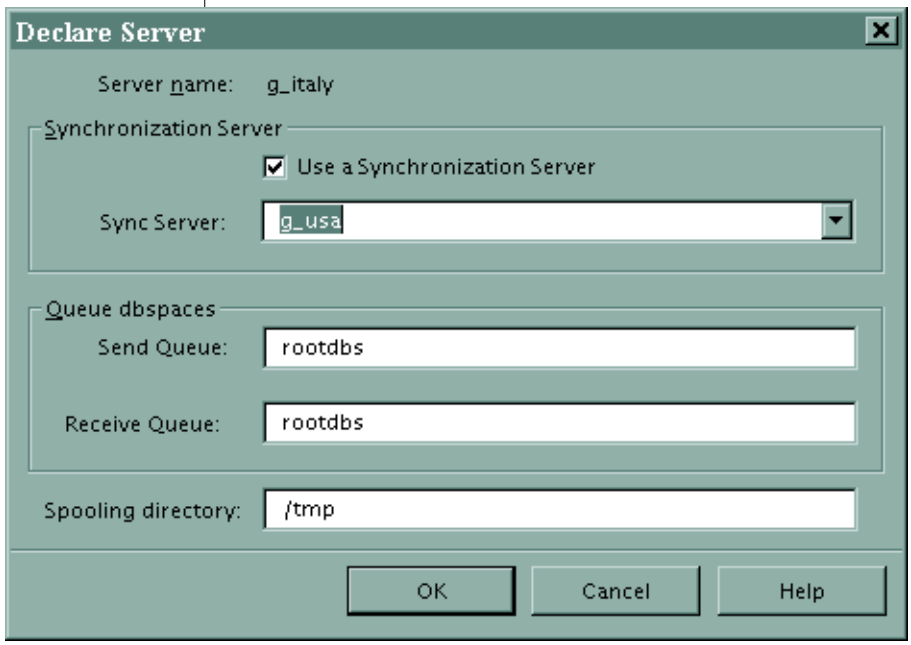



***Figure 3-10***
*Declare the Second Replication Server*

After you declare the **italy** database server as a replication server, the main ERM display shows the replication servers in your replication environment, as Figure 3-11 shows.

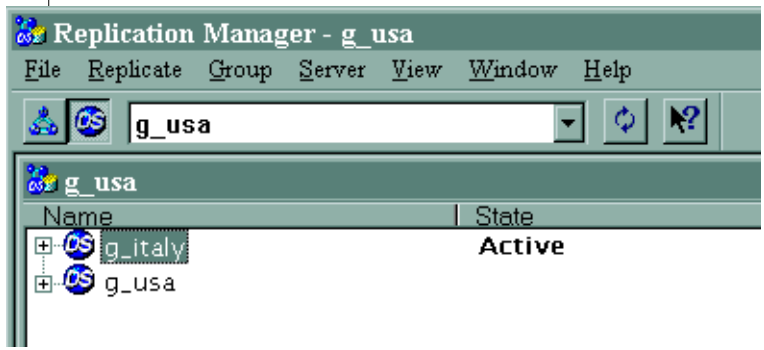



***Figure 3-11***
*List of Replication Servers*

## Creating the Primary-Target Replicate

This section creates a replicate that causes any change to the **call_type** table of the **stores** database on the **usa** database server to be replicated to the **call_type** table on the other database servers. This replicate is equivalent to the replicate in "Update-Anywhere Example Using the CLU" on page 3-9, except that the table replicated is **call_type** instead of **manufact**.

### Choosing a Name for the Replicate

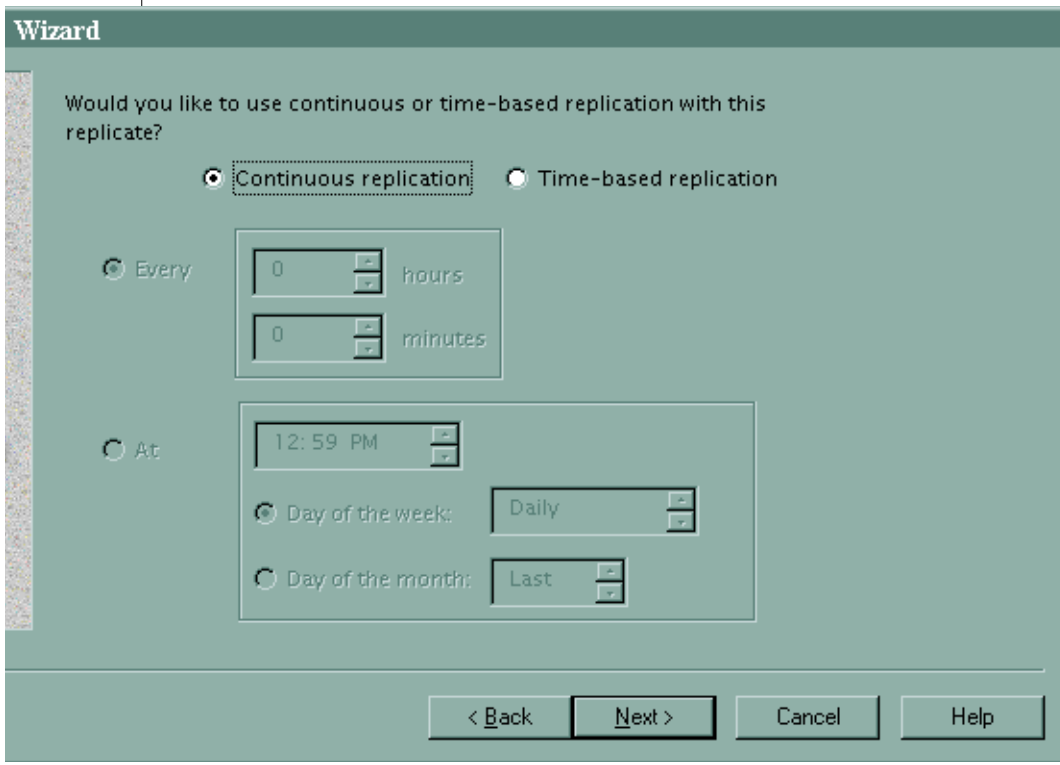
Select **Define Replicate** from the Replicate menu. ERM displays the introductory page of the Define Replicate wizard. Click **Next**.

On the next page, enter a name for the replicate, **repl3**. Click **Next**.

### Specifying the Replication Frequency

When a change is made to a table that is participating in a replicate, Enterprise Replication can propagate the change either immediately or at fixed intervals. The default behavior propagates the change immediately. On the wizard page that Figure 3-12 shows, leave the frequency set at Continuous Replication. Click **Next**.

***Figure 3-12***
*Specify Replication Frequency*

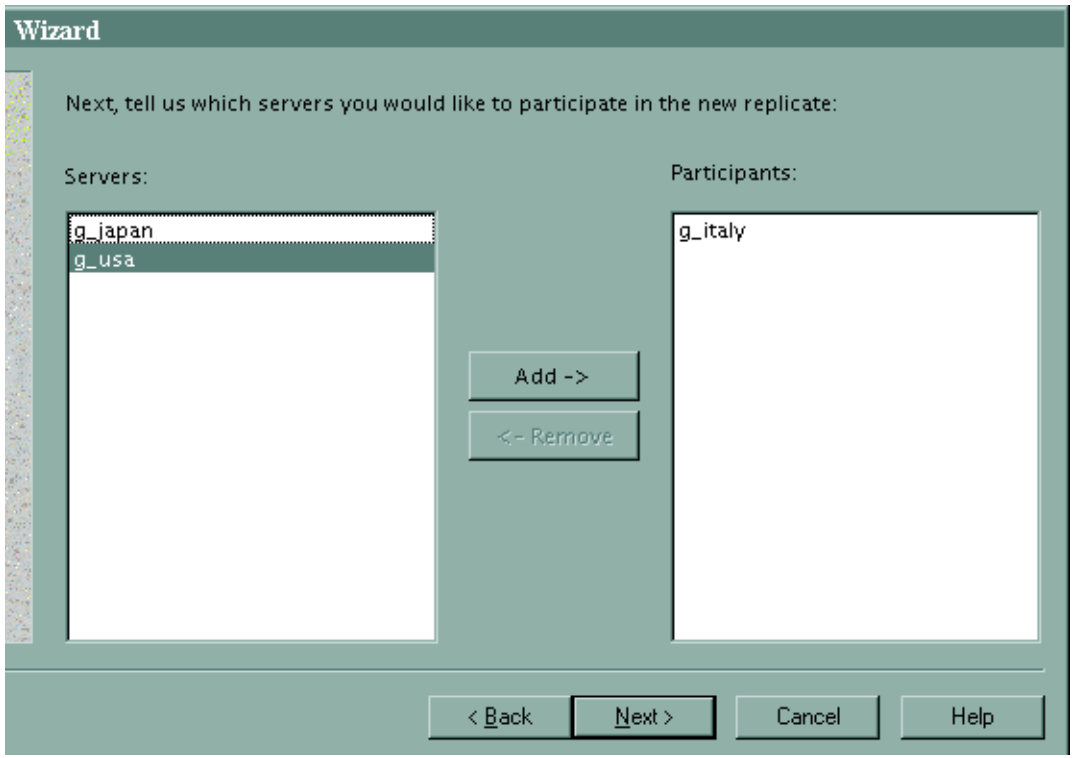


### Selecting the Participants

The wizard page in Figure 3-13 on page 3-28 lets you specify the database servers for this replication. Select **g_italy** and click **Add**.

Then select **g_usa**, click **Add,** and then click **Next**.

***Figure 3-13***
*Select Servers for the Replicate*

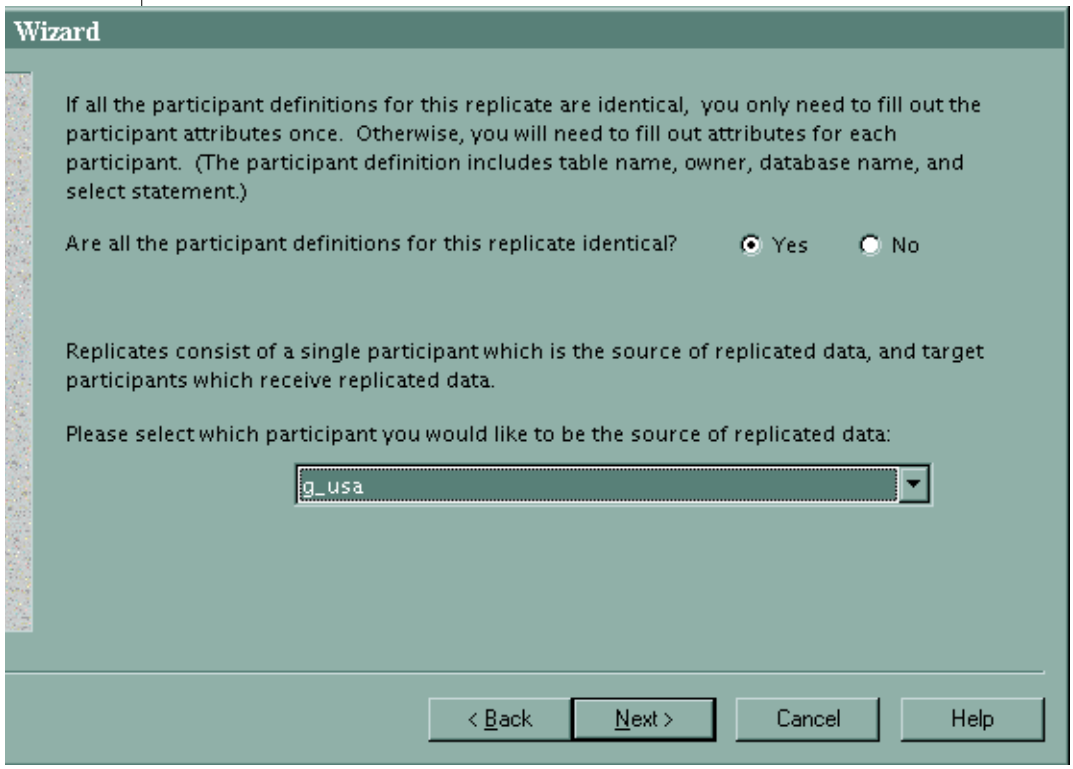


### Selecting the Primary

Enterprise Replication replicates changes from the primary participant to the secondary participant. On the wizard page in Figure 3-14 on page 3-29, make sure that you select the **Yes** option button.

In the list box, select **g_usa** and click **Next**.

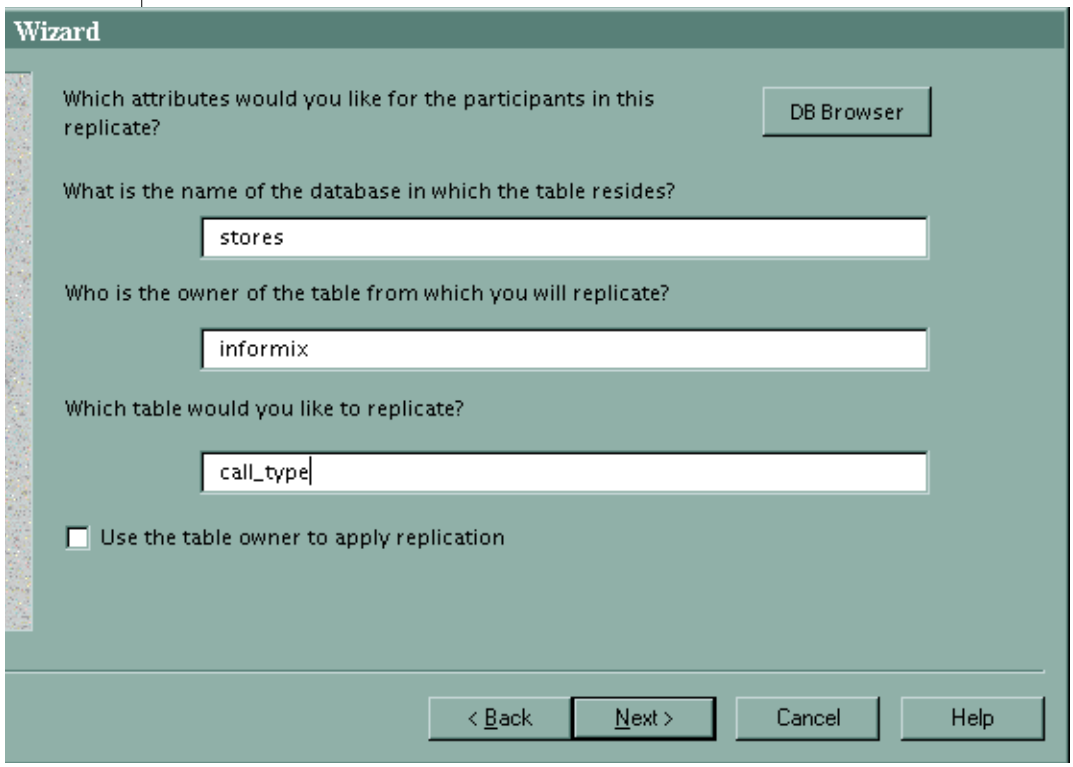
***Figure 3-14***
*Select the Primary Participant*


Wizard
If all the participant definitions for this replicate are identical, you only need to fill out the participant attributes once. Otherwise, you will need to fill out attributes for each participant. (The participant definition includes table name, owner, database name, and select statement.)
Are all the participant definitions for this replicate identical?
Yes
No
Replicates consist of a single participant which is the source of replicated data, and target participants which receive replicated data.
Please select which participant you would like to be the source of replicated data:
g_usa
< Back
Next >
Cancel
Help

### Selecting a Table

The wizard page in Figure 3-15 lets you select the table to replicate. Enter the name of the database, the table owner (the login name you used when you created the table), and the name of the table. Click **Next**.

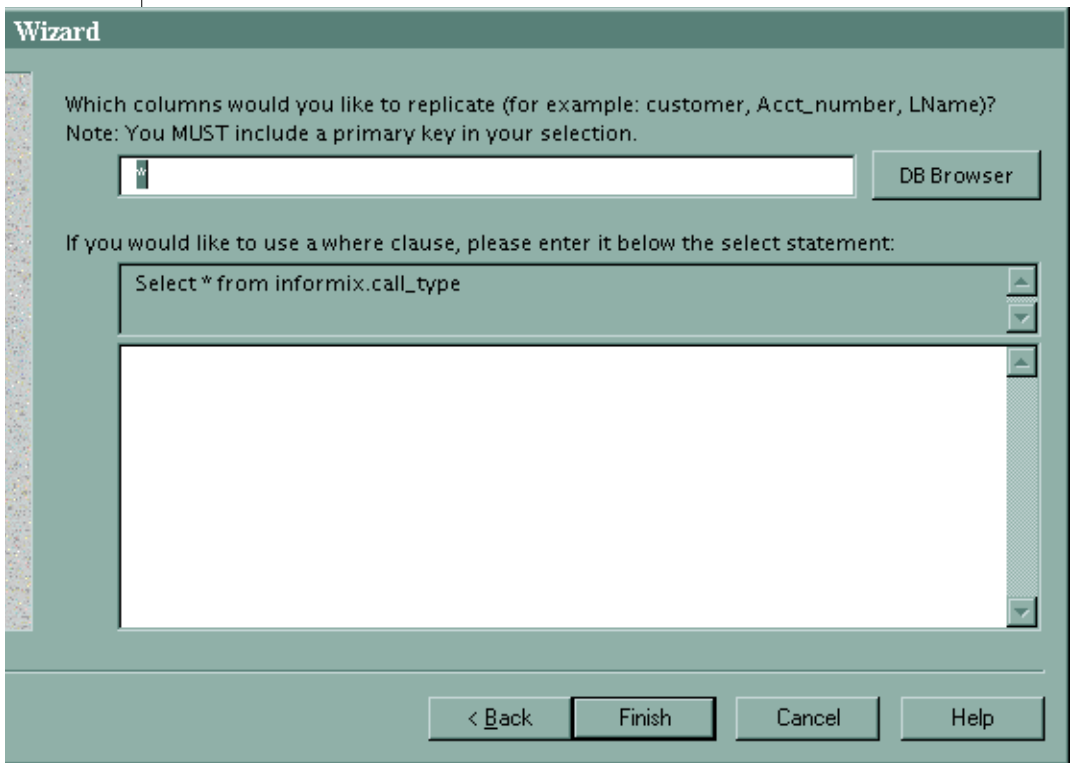
***Figure 3-15***
*Select a Table for Replication*


Wizard
Which attributes would you like for the participants in this replicate?
DB Browser
What is the name of the database in which the table resides?
stores
Who is the owner of the table from which you will replicate?
informix
Which table would you like to replicate?
call_type
Use the table owner to apply replication
< Back
Next >
Cancel
Help

### Selecting the Columns

The wizard page in Figure 3-16 lets you specify which columns of the table should be replicated. The **SELECT *** statement causes all columns of the **call_type** table to be replicated. There is no WHERE clause, so all changes to **call_type** will be replicated. Click **Finish**.

***Figure 3-16***
*Prepare Select Statement*


Wizard
Which columns would you like to replicate (for example: customer, Acct_number, LName)?
Note: You MUST include a primary key in your selection.
DB Browser
If you would like to use a where clause, please enter it below the select statement:
Select * from informix.call_type
< Back
Finish
Cancel
Help

### Summary

The Replicate Summary page in Figure 3-17 lets you review the attributes of the replicate before ERM actually creates the replicate. Read the summary to verify that it shows the choices that you made. If you are not satisfied, click **Cancel** to revise your replicate. If the summary is correct, click **Create**.

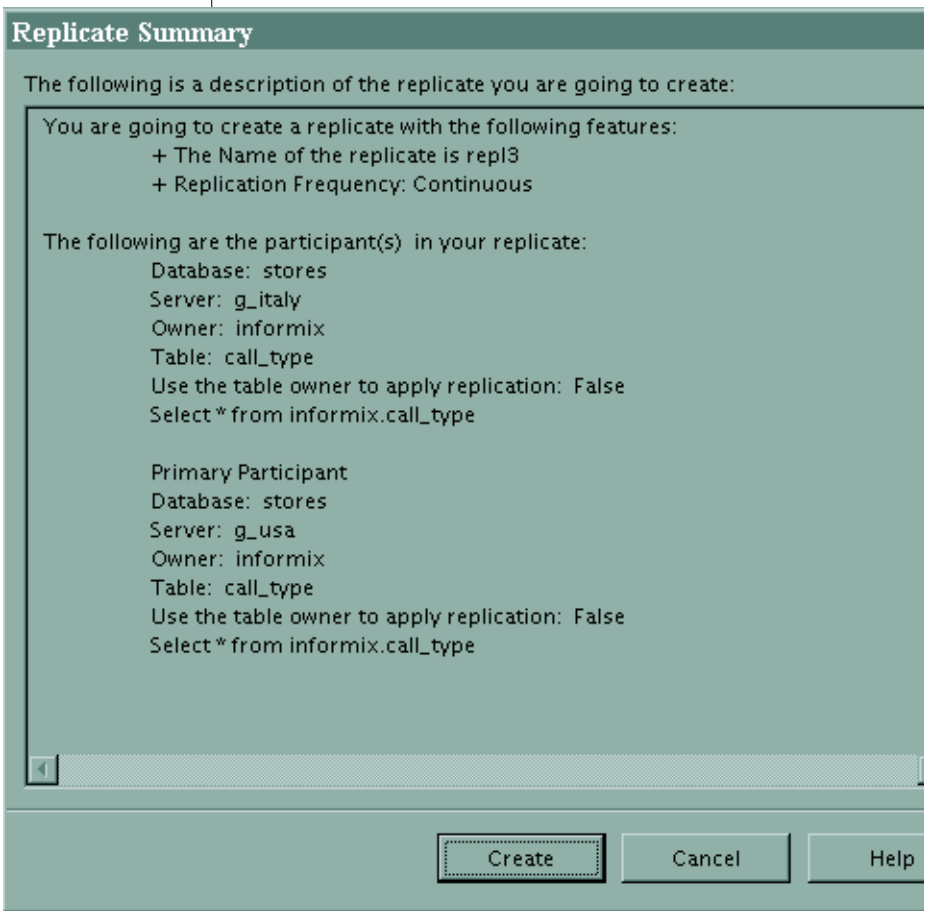

Replicate Summary
The following is a description of the replicate you are going to create:
You are going to create a replicate with the following features:
+ The Name of the replicate is repl3
+ Replication Frequency: Continuous
The following are the participant(s) in your replicate:
Database: stores
Server: g_italy
Owner: informix
Table: call_type
Use the table owner to apply replication: False
Select * from informix.call_type
Primary Participant
Database: stores
Server: g_usa
Owner: informix
Table: call_type
Use the table owner to apply replication: False
Select * from informix.call_type
Create
Cancel
Help

***Figure 3-17***
*Summary of repl3*

## Viewing the Replicate

When you click **Create** on the Replicate Summary page, as Figure 3-17 on page 3-32 shows, ERM creates the replicate and returns to the main ERM display. Click the replicate icon, , in the upper-left part of the display. ERM lists the replicates that you defined. (If you reinitialized your database servers in "Removing Previously Defined Replication Servers" on page 3-17, the list has only one replicate, **repl3**.)

Click the plus (+) beside **repl3** to list the database servers that are participants in the replicate. Figure 3-18 shows the participants in **repl3**.

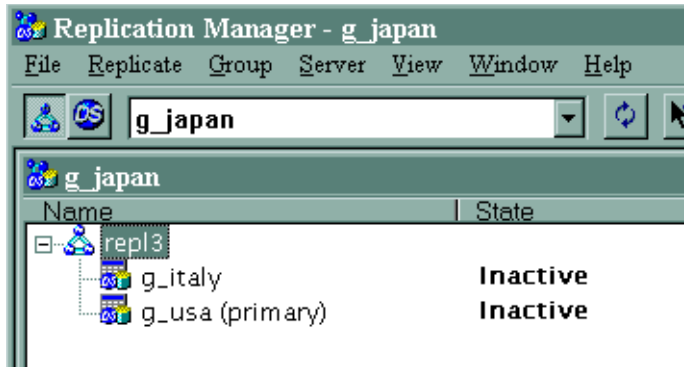



***Figure 3-18***
*Display of repl3*

## Activating the Replicate

Figure 3-18 shows that both **g_italy** and **g_usa** are inactive in **repl3**. Although you have defined the replicate, no changes will be replicated until you make the replicate active.

To activate the replicate, select **repl3** and then choose **Start Replicate** from the Replicate menu, as Figure 3-19 shows.

***Figure 3-19***
*Replicate Menu*

When you select Start Replicate, ERM displays the Start Replicate dialog box in Figure 3-20. Click **Select All** and then click **OK**.

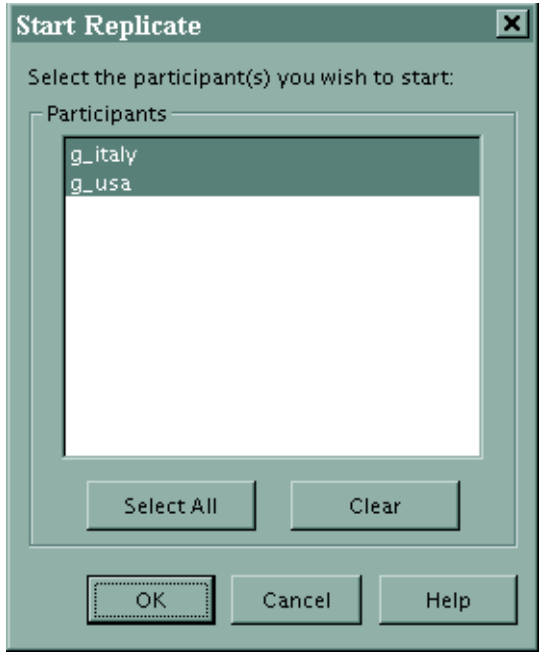



***Figure 3-20***
*Start Replicate Dialog Box*

ERM sets the state of all participants in **repl3** to Active, as Figure 3-21 shows.

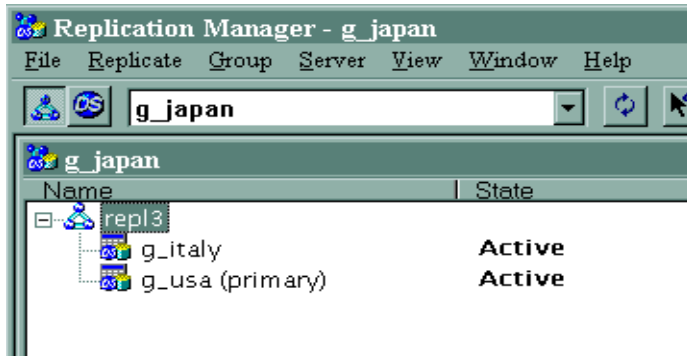



***Figure 3-21***
*State of Participants in repl3*

## Cause a Replication

You can now modify the **call_type** table and observe the replication. For example, to insert a line into the **call_type** table on **usa**, you can use DB-Access, as follows:

```
INSERT INTO stores@usa:stock VALUES (401, “PRC”, “ski boots”,
200.00, “pair”, “pair”);
```

Observe the change on the **italy** database server:

```
SELECT * from stores@italy:stock;
```

## Do Not Cause a Replication

Changes to the secondary participant (**italy**) in **repl3** do not replicate to the primary (**usa**). For example, to change a value in the **call_type** table on **italy**, you can use DB-Access, as follows:

```
UPDATE stores@italy:call_type SET code_descr = 'wrong size'
WHERE call_code = 'C';
```

Verify that the change has *not* replicated to the **call_type** table on **usa**:

```
SELECT * from stores@usa:call_type;
```

# Update-Anywhere Example Using ERM

This example uses ERM to perform the same simple *update-anywhere* replication that the "Update-Anywhere Example Using the CLU" on page 3-9 shows.

## Changing to Expert Mode

The only replication option available in nonexpert mode is primary-target replication. You must switch to expert mode to do update-anywhere replication. For instructions on how to switch between expert and nonexpert modes, refer to "Choosing Nonexpert Mode" on page 3-23.

## Creating an Update-Anywhere Replicate

This section creates a replicate that causes replication of any change to the **state** table of the **stores** database on *any* of the database servers in the replicate. This replicate is equivalent to the replicate in "Update-Anywhere Example Using the CLU" on page 3-9, except that the table replicated is **state** instead of **stock**.

### *Choosing a Replicate Name and Type*

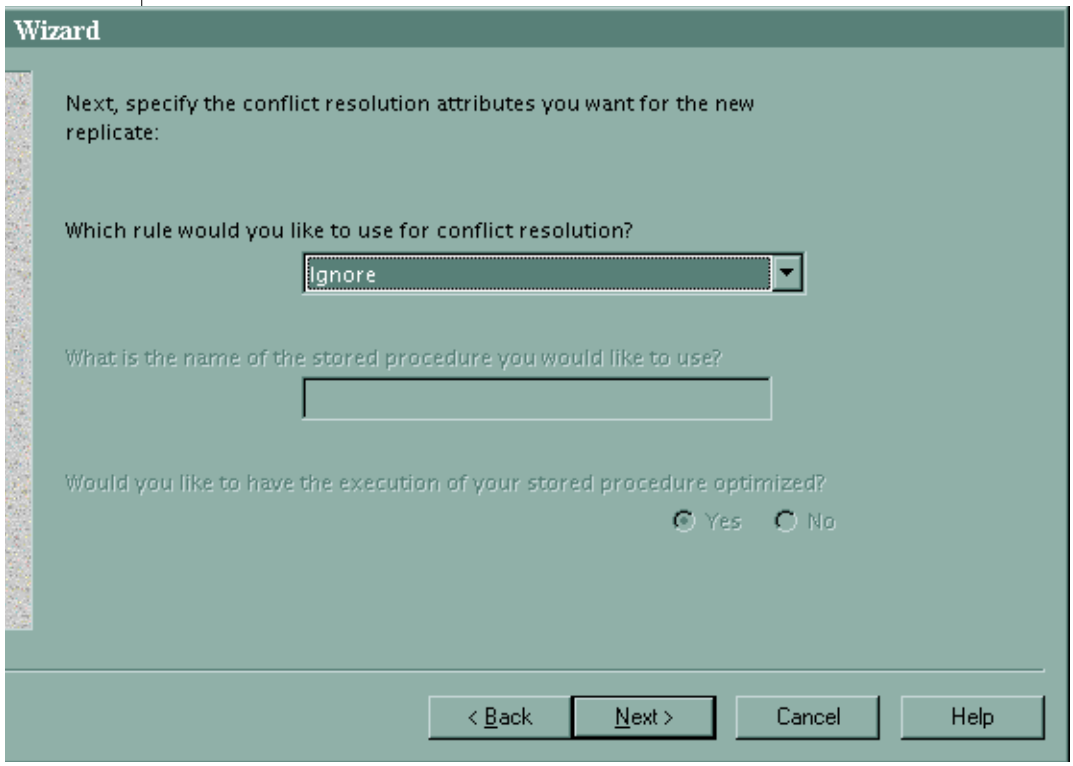
Select **Replicate→Define Replicate**. ERM displays the introductory page of the Define Replicate wizard. Click **Next**.

On the next page, enter a name for the replicate, **repl4**. Click the **Update-Anywhere** option button and then click **Next**.

### Specifying a Conflict Rule

In update-anywhere replication, conflicts can occur when two users update the same row at the same time. Enterprise Replication provides conflict-resolution rules for resolving conflicts. The display in Figure 3-22 lets you specify the conflict rule. For this example, choose the default, which is the ignore rule. Click **Next**.

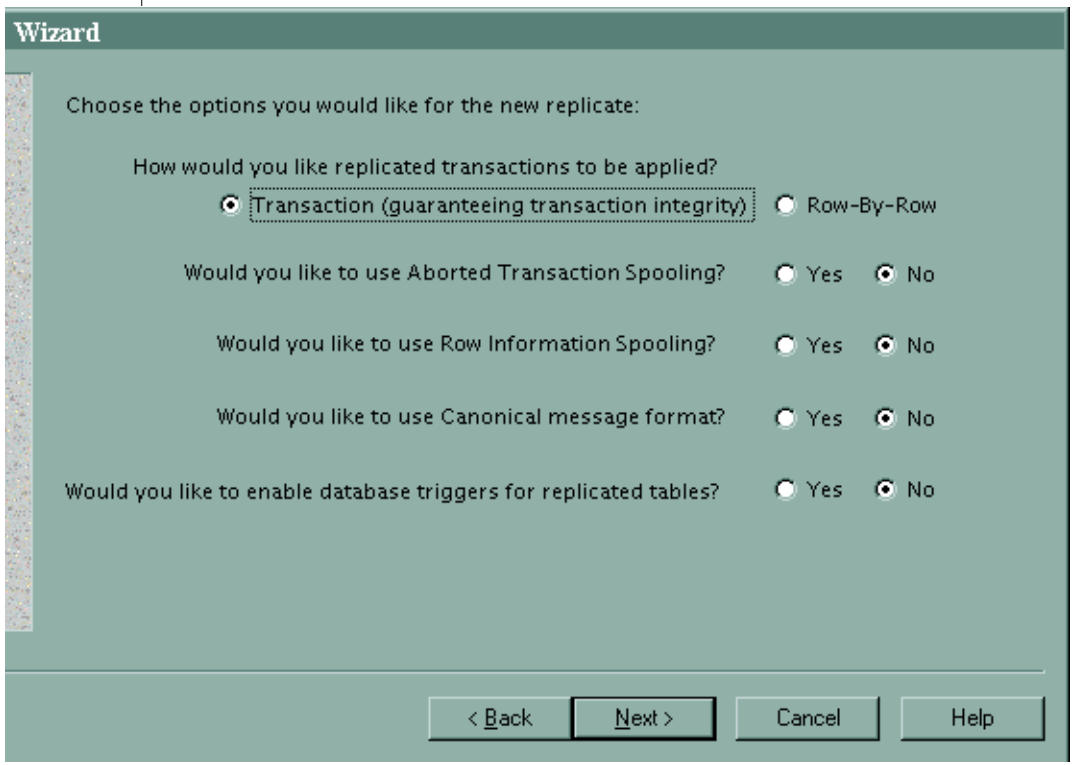
***Figure 3-22***
*Select Conflict Resolution Rule*


Wizard
Next, specify the conflict resolution attributes you want for the new replicate:
Which rule would you like to use for conflict resolution?
Ignore
What is the name of the stored procedure you would like to use?
Would you like to have the execution of your stored procedure optimized?
Yes
No
< Back
Next >
Cancel
Help

## Specifying the Replication Options

The wizard page in Figure 3-23 lets you specify options for the replicate. You can leave the options set to the defaults that the wizard suggests. Click **Next**.

***Figure 3-23***
*Replication Options*




## Specifying the Frequency

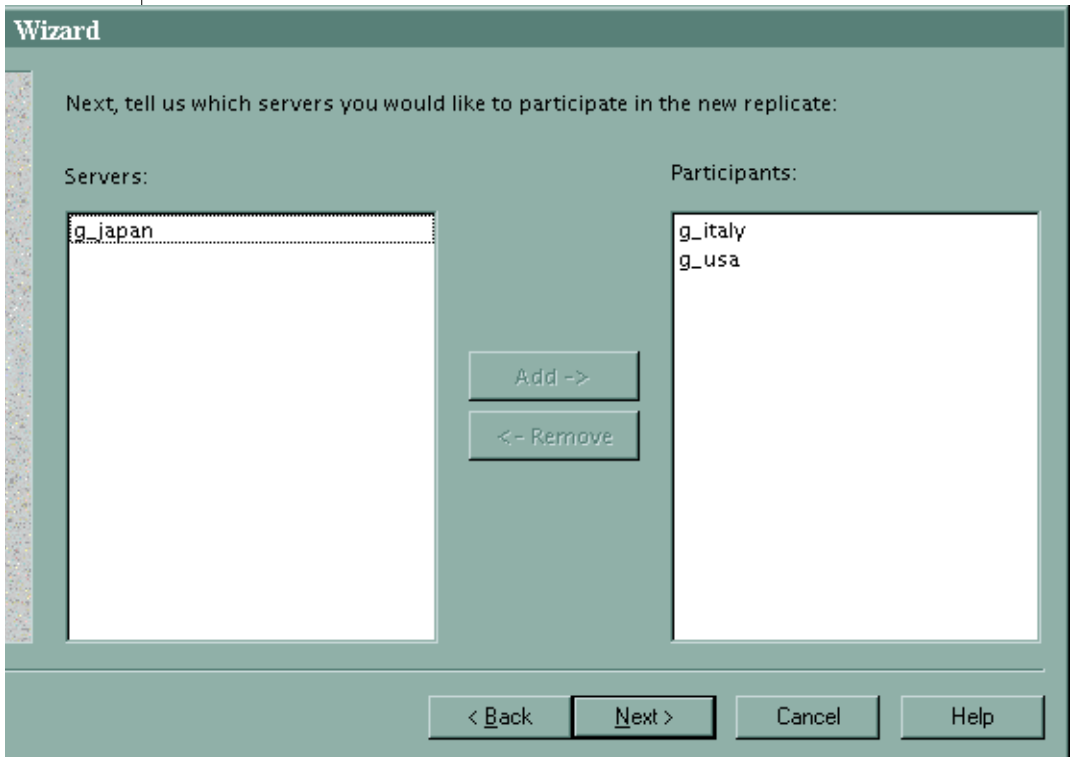
When a change is made to a table that is participating in a replicate, Enterprise Replication can propagate the change either immediately or at fixed intervals. The choices for frequency are the same as those that Figure 3-12 on page 3-27 shows, which are continuous or time-based with continuous being the default. Do not change the default settings. Click **Next**.

### Selecting the Participants

On the Select Participants wizard page, select the **italy** and **usa** servers and then click **Add**. (You could also select **japan**, but one purpose of this exercise is to add **japan** to an already-defined replicate.)

Figure 3-24 shows the completed dialog box. Click **Next**.

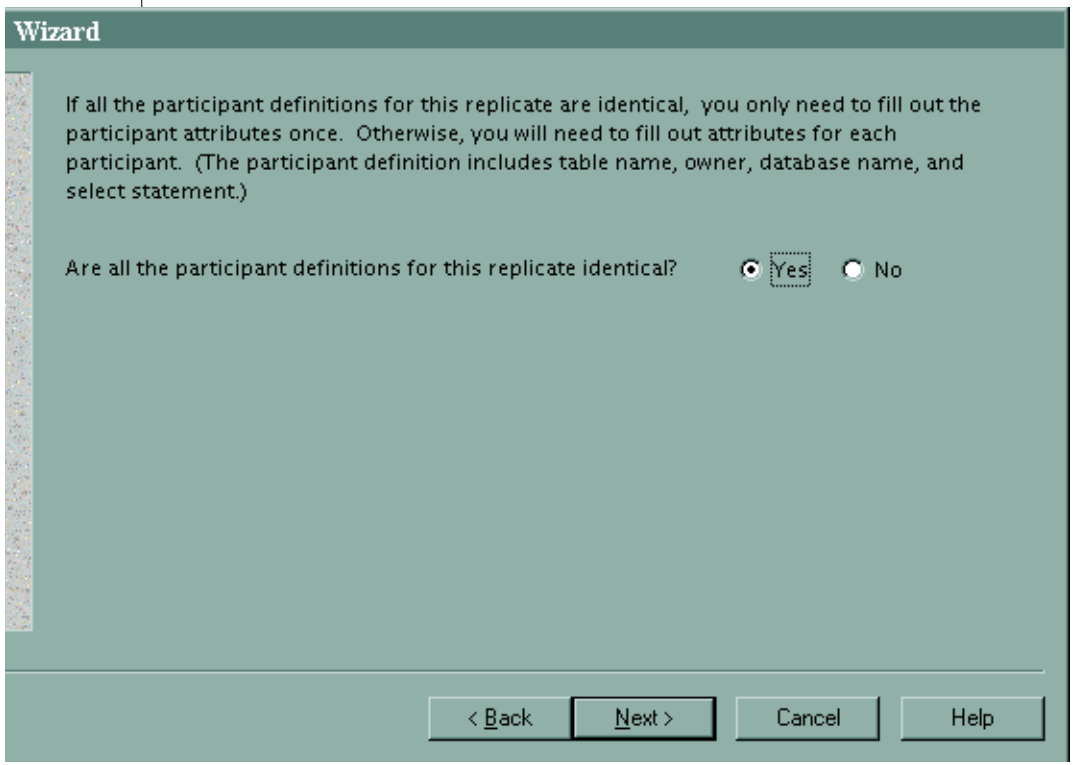
***Figure 3-24***
*Select Participants*



### Participants Identical

The wizard page in Figure 3-25 asks if the participant definitions (that is, the database name, owner, and table name) are identical. Click the **Yes** option button and then click **Next**.

***Figure 3-25***
*Identical Replicate Definitions*


Wizard
If all the participant definitions for this replicate are identical, you only need to fill out the participant attributes once. Otherwise, you will need to fill out attributes for each participant. (The participant definition includes table name, owner, database name, and select statement.)
Are all the participant definitions for this replicate identical?
Yes
No
< Back
Next >
Cancel
Help

### Selecting a Table

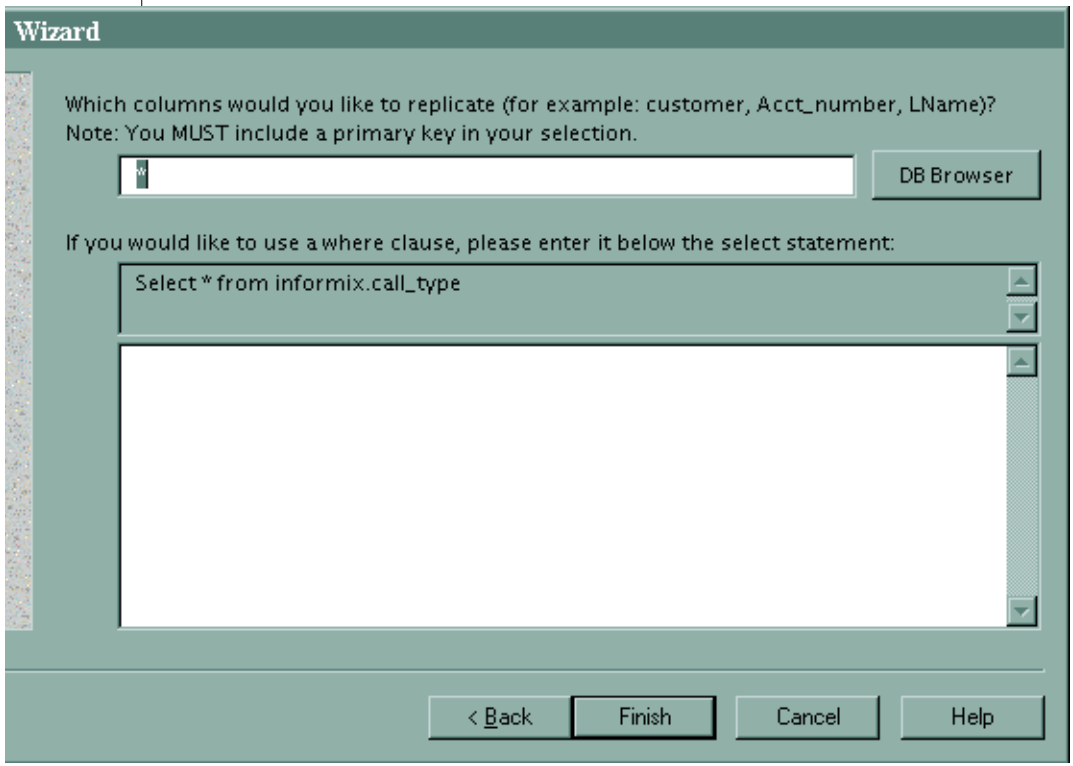
The wizard pages for selecting a table and columns for update-anywhere replication are the same as the corresponding wizard pages for primary-target replication.

The wizard page in Figure 3-15 on page 3-30 lets you select the table to replicate. Enter the name of the database, the table owner (the login name you used when you created the table), and the name of the table for this replicate (**state**). Click **Next.**

### Selecting the Columns

The wizard page in Figure 3-26 lets you specify which columns of the table should be replicated. The **SELECT *** statement causes all columns of the table to be replicated. There is no WHERE clause, so all changes to **state** table will be replicated. Click **Finish**.

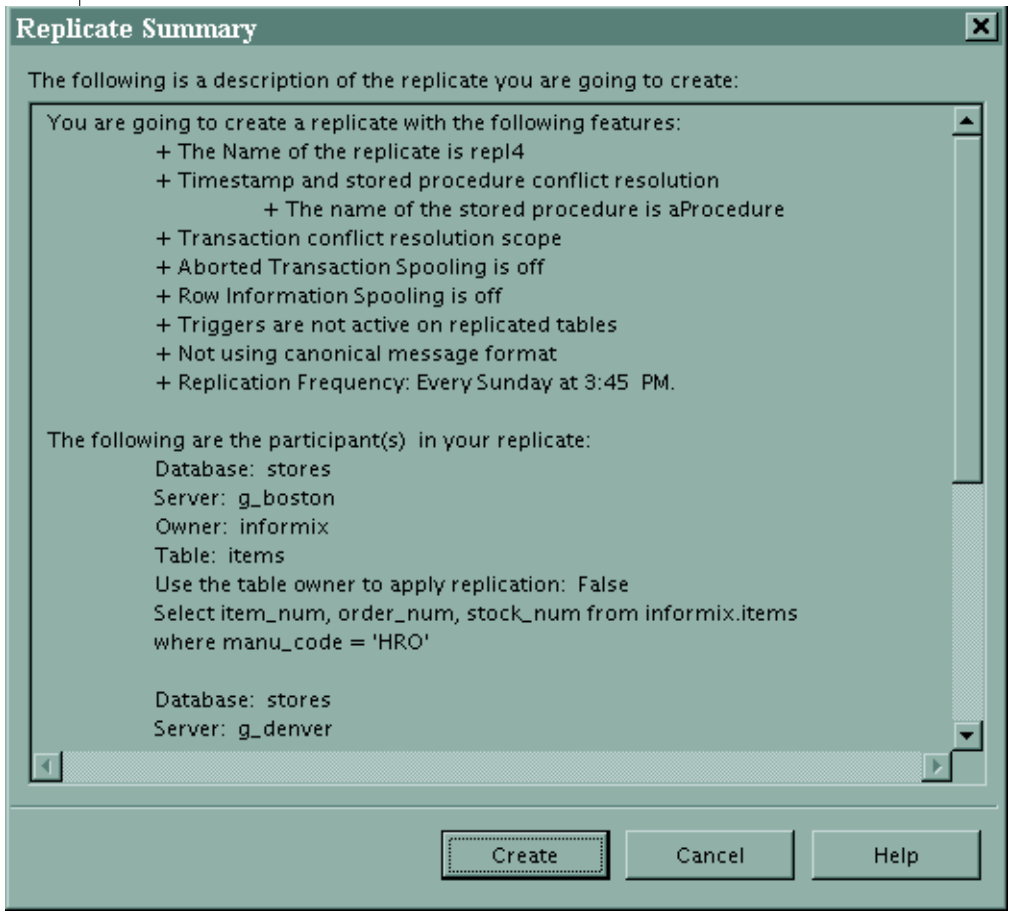
***Figure 3-26***
*Prepare Select Statement*


Wizard
Which columns would you like to replicate (for example: customer, Acct_number, LName)?
Note: You MUST include a primary key in your selection.
DB Browser
If you would like to use a where clause, please enter it below the select statement:
Select * from informix.call_type
< Back
Finish
Cancel
Help

### Summary

The Replicate Summary page in Figure 3-27 lets you review the attributes of the replicate before ERM actually creates the replicate. Read the summary to verify that it shows the choices that you made. Use the scroll bar to see the rest of the display. If the summary is correct, click **Create**.

***Figure 3-27***
*Summary of repl4*


Replicate Summary
The following is a description of the replicate you are going to create:
You are going to create a replicate with the following features:
+ The Name of the replicate is repl4
+ Timestamp and stored procedure conflict resolution
+ The name of the stored procedure is aProcedure
+ Transaction conflict resolution scope
+ Aborted Transaction Spooling is off
+ Row Information Spooling is off
+ Triggers are not active on replicated tables
+ Not using canonical message format
+ Replication Frequency: Every Sunday at 3:45 PM.
The following are the participant(s) in your replicate:
Database: stores
Server: g_boston
Owner: informix
Table: items
Use the table owner to apply replication: False
Select item_num, order_num, stock_num from informix.items
where manu_code = 'HRO'
Database: stores
Server: g_denver
Create
Cancel
Help

## Viewing the Replicate

When you click **Create** on the Replicate Summary page, as Figure 3-27 on page 3-43 shows, ERM creates the replicate and returns to the main ERM display, as Figure 3-28 shows. Click the replicate icon in the upper-left part of the display. ERM lists the replicates that you defined.

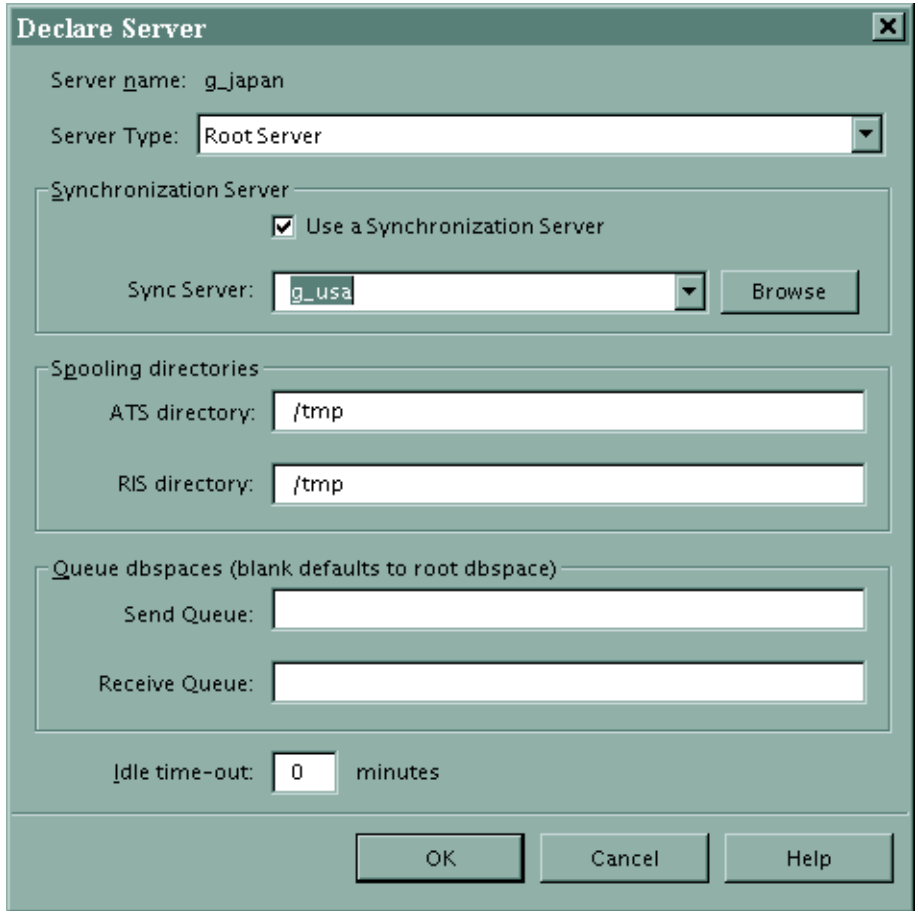
***Figure 3-28***
*List of Replicates*

## Defining a Replication Server on japan

This update-anywhere replicate should include changes made on the **japan** database server. Select **g- japan** from the Server list box (see Figure 3-9 on page 3-24). ERM asks if you want to start Enterprise Replication on **g-japan**. Click **Yes**.

On the wizard page in Figure 3-29, select **Root Server** in the Server Type dialog box and **g_usa** in the Sync Server dialog box. Click **OK**.

***Figure 3-29***
*Add japan as a Replication Server*



## Adding a Participant to repl4

To add **japan** as a participant to **repl4**, select **repl4** and then choose **Replicate→Add Participant(s)**. You can select **repl4** on the replicate display, as Figure 3-30 shows, or the server display, as Figure 3-31 shows.

***Figure 3-30***
*Select from Replicate Display*

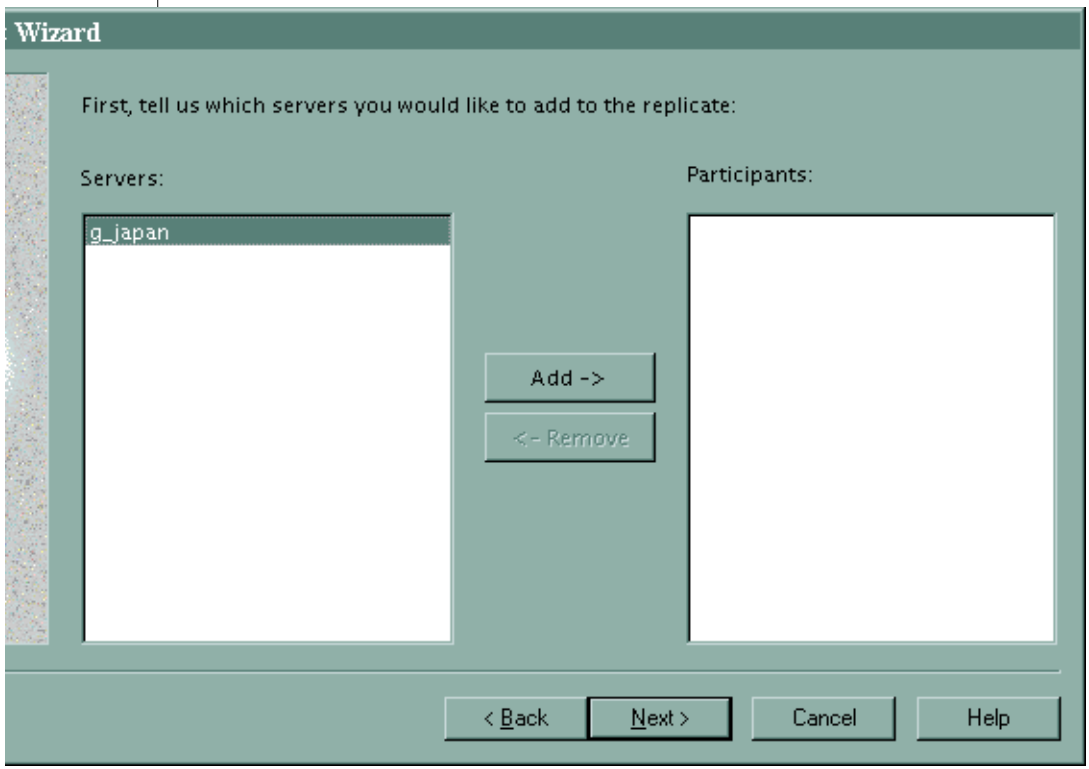
***Figure 3-31***
*Select from Server Display*

### *Selecting a New Participant*

The Add Participant wizard displays an introductory page. Click **Next**.

The next wizard page, Figure 3-32, asks which servers you want to add to the replicate. Select **g_japan**, click **Add,** and then click **Next**.

***Figure 3-32***
*Select New Participant*



### Specifying a Table and Columns

The next two wizard pages ask you to specify the participant attributes and the columns to replicate. Use the discussions in "Selecting a Table" on page 3-30 and "Selecting the Columns" on page 3-31 as guides for completing the pages.

When you click **Finish**, ERM adds the new participant (**g-japan**) to **repl4**. ERM does not show a summary page before it creates the new participant.

## Activating the Replicate

On the main ERM display, click the replicate icon and then click the plus (+) beside **repl4** to display the database servers that are participants in the replicate. The display, as Figure 3-33 indicates, shows that all participants in the replicate are inactive.

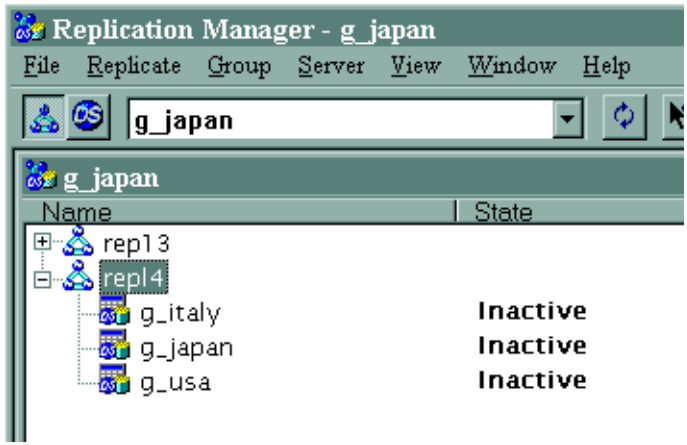



***Figure 3-33***
*Inactive Participants*

Follow the procedure described in "Activating the Replicate" on page 3-34 to activate all participants in **repl4**.

## Cause a Replication

You can now modify the **state** table and observe the replication. For example, to insert a line into the **state** table on **usa,** you can use DB-Access, as follows:

```
INSERT INTO stores@usa:state VALUES (AT, "Atlantis");
```

Observe the change on the **italy** and **japan** database servers:

```
SELECT * from stores@italy:state;
SELECT * from stores@japan:state;
```

To insert a line into the **state** table on **japan**, use DB-Access, as follows:

```
INSERT INTO stores@japan:state VALUES (BR, "Brigadoon");
```

Verify that the new line was replicated to **usa** and **italy**.

# Hierarchy Example Using ERM

This example uses ERM to add a replication tree to the fully-connected environment of the **usa**, **italy**, and **japan** replication servers. Figure 3-1 on page 3-13 shows the hierarchy that this example will build.

## Deleting the Previously-Defined Servers

If you did not delete the **denver**, **boston**, and **miami** servers in "Removing Previously Defined Replication Servers" on page 3-17, you must delete them now. For each server, issue the following command, where *server_group_name* is **g_denver**, **g_boston**, or **g_miami**:

```
cdr delete server server_group_name
cdr delete server -c server_group_name server_group_name
```

## Defining the Nonroot Replication Servers

Nonroot replication servers are continuously connected to their parent servers but are not connected to the other root servers.

### Defining the boston Nonroot Server

To add **boston** as a nonroot replication server, select **boston** from the Server list box (see Figure 3-9 on page 3-24). When ERM asks if you want to define **boston** as a replication server, click **Yes**.

On the Declare Server wizard page, select **Nonroot Server** in the Server Type dialog box. Select **g_boston** in the Parent Server dialog box.

Leave the other values set to their defaults. If you want to view the completed wizard page, see Figure 3-6 on page 3-22. Click **OK**.

### Defining the denver Nonroot Server

Follow the procedure in the previous section to declare **denver** as a nonroot replication server.

## Defining a Leaf Replication Server

A leaf server might have either a root or nonroot server as its parent. A leaf server cannot be the parent of another replication server.

To add **miami** as a leaf server, select **miami** from the Server list box. When ERM asks if you want to define **miami** as a replication server, click **Yes**.

On the Declare Server wizard page, select **Leaf Server** in the Server Type dialog box. Select **g_usa** in the Parent Server dialog box.

Leave the other values set to their defaults. If you want to view the completed wizard page, see Figure 3-6 on page 3-22. Click **OK**.

## Displaying the Hierarchy

After ERM defines the **miami** leaf server, the display returns to the main ERM display. Click the database server icon to show a list of replication servers. The display changes depending on which server you select from the Server list box. Try selecting **g_italy**, **g_denver**, and **g_miami** to see how the display changes. None of the lists indicates the hierarchy of the servers.

To see the replication hierarchy, select **Server→Hierarchical Routing**. The Hierarchical Routing view in Figure 3-34 shows that **boston** and **denver** are children of the **usa** root server and that **miami** is a child of the **boston** nonroot server.

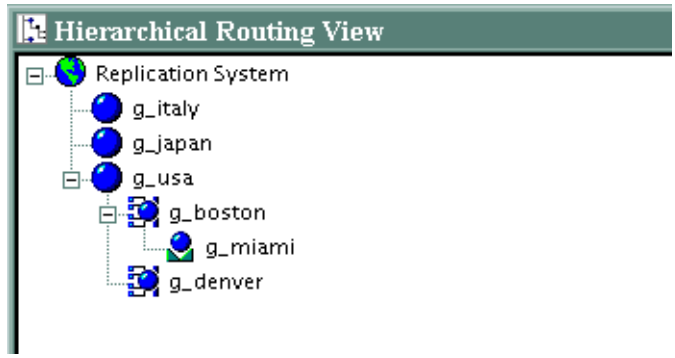



***Figure 3-34***
*Hierarchical Routing View*

## Viewing the Server Properties

You can also see parent/child relationships when you look at server properties. Select the **miami** server from either the Hierarchical Routing View window or from the main ERM display. Then select **Server→Server Properties**. Select the Routing tab in the Server Properties window. The display shows that **miami** is a leaf and its parent is **boston**, as in Figure 3-35.

***Figure 3-35***
*Server Properties Dialog Box*

Section II

# Preparing for Enterprise Replication


Chapter 4 Selecting Enterprise-Replication Systems

Chapter 5 Setting Up Your Replication Environment

Chapter 6 Preparing Data for Replication

Chapter

# 4

# Selecting Enterprise-Replication Systems


In This Chapter . . . . . 4-3

Primary-Target Replication System . . . . . 4-3
- Data Dissemination . . . . . 4-4
- Data Consolidation . . . . . 4-5
- Workload Partitions . . . . . 4-6
- Workflow Replication . . . . . 4-7

Factors in Primary-Target Systems . . . . . 4-7
- Selecting Data for Primary-Target Replication . . . . . 4-8
- Administering the Replication System . . . . . 4-8
- Planning for Capacity . . . . . 4-8
- Planning for High Availability . . . . . 4-8

Update-Anywhere Replication System . . . . . 4-9
- Facts to Consider . . . . . 4-9
  - Selecting Data for Replication . . . . . 4-9
  - Administering the Replication System . . . . . 4-10
  - Delivering Consistent Information . . . . . 4-10
  - Planning for Capacity . . . . . 4-10
  - Planning for High Availability . . . . . 4-10

## In This Chapter

This chapter describes Enterprise Replication system types and discusses the trade-offs associated with performance and data availability. Enterprise Replication supports the following replication types:

- Primary target
- Update anywhere

## Primary-Target Replication System

In the primary-target replication system, the flow of information is in one direction. That is, the primary always sends data to the target. The target does not send data to the primary. In this type of replication system, no conflict detections or rules to resolve conflicts are necessary. Primary-target replication systems support the following business models:

- Data dissemination
- Data consolidation
- Workload partitions
- Workflow replication

## Data Dissemination

Data dissemination supports a business need where data is updated in a central location and then replicated to read-only sites. This method of distribution can be particularly useful for on-line transaction processing (OLTP) systems where data is required at several sites, but because of the large amounts of data, read / write capabilities at all sites would cripple the performance of the application. Figure 4-1 illustrates data dissemination.

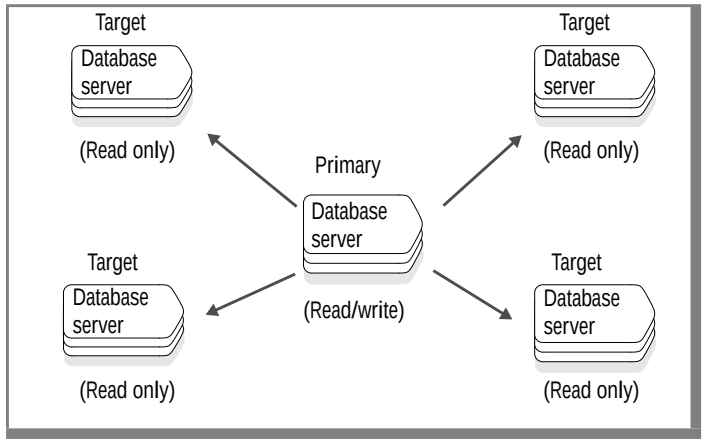



***Figure 4-1***
*Data Dissemination in a Primary-Target Replication System*

## Data Consolidation

As businesses reorganize to become more competitive, many choose to consolidate data into one central database server. Data consolidation allows the migration of data from several database servers to one central database server. In Figure 4-2, the remote locations have read / write capabilities while the central database server is read only.

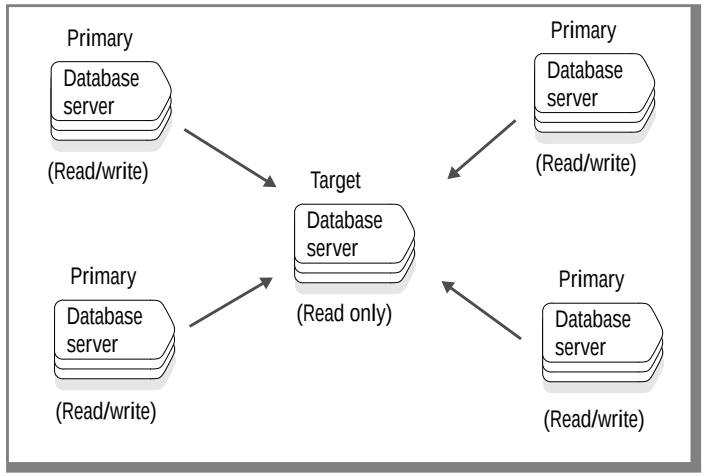



***Figure 4-2***
*Data Consolidation in a Primary-Target Replication System*

Businesses can also use data consolidation to off-load OLTP data for decision support (DSS) analysis. For example, data from several OLTP systems can be replicated to a DSS system for read-only analysis. Pay close attention to the configuration of the tables from which data is replicated to ensure that each primary key is unique among the multiple primary database servers.

## Workload Partitions

Workload partitioning gives businesses the flexibility to assign data ownership at the table-partition level, rather than within an application. Figure 4-3 illustrates workload partitioning.

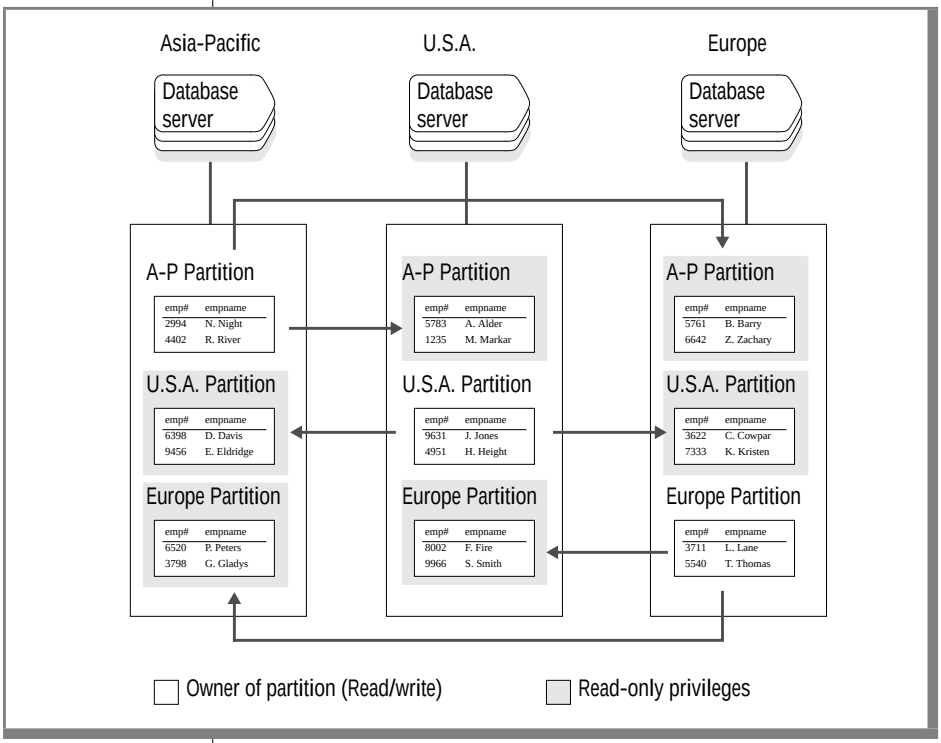



***Figure 4-3***
*Workload Partitioning in a Primary-Target Replication System*

In Figure 4-3, the replication model matches the partition model for the **employee** tables. The Asia-Pacific computer owns the partition and can therefore update, insert, and delete employee records for personnel in its region. The changes are then propagated to the U.S. and European regions. Asia-Pacific can query or read the other partitions locally but cannot update those partitions locally. This strategy applies to other regions as well.

## Workflow Replication

Unlike the data dissemination model, in a workflow-replication system, the data moves from site to site. Each site processes or approves the data before sending it on to the next site.

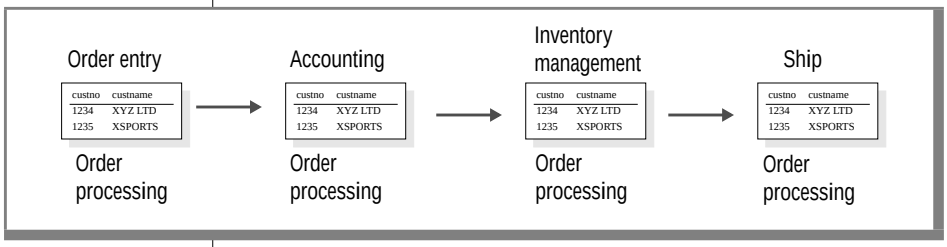



***Figure 4-4*** *A Workflow-Replication System Where Update Authority Moves From Site to Site*

Figure 4-4 illustrates an order-processing system. Order processing typically follows a well-ordered series of steps: orders are entered, approved by accounting, inventory is reconciled, and the order is finally shipped.

In a workflow-replication system, application modules can be distributed across multiple sites and databases. Data can also be replicated to sites that need read-only access to the data (for example, if order-entry sites want to monitor the progress of an order).

A workflow-replication system, like the primary-target replication system, only allows unidirectional updates. Many facts that you need to consider for a primary-target replication system should also be considered for the workflow-replication system.

However, unlike the primary-target replication system, availability can become an issue if a database server goes down. The database servers in the workflow-replication system rely on the data updated at a previous site. Consider this fact when you select a workflow-replication system.

# Factors in Primary-Target Systems

The following sections describe some of the factors to consider when you select a primary-target replication system.

## Selecting Data for Primary-Target Replication

The requirements for unique primary keys limit the use of the SERIAL and SERIAL8 data types in schema that participate in a replication system. You can replicate from a serial data type column, but the destination column must be an integer. When Enterprise Replication applies rows in this situation, it sets the destination integer column to the value assigned at the source database server.

## Administering the Replication System

Primary-target replication systems are the easiest to administer because all updates are unidirectional and therefore, no conflict detection or resolution rules are required.

## Planning for Capacity

All replication systems require you to plan for capacity changes. For more information, see Chapter 5, “Setting Up Your Replication Environment.”

## Planning for High Availability

In the primary-target replication system, if a target database server goes down or the network goes down, Enterprise Replication continues to log information for the database server until it becomes available. If a database server is unavailable for some time, you might want to remove the database server from the replication system. If the unavailable database server is the read / write database server, you will need to plan a course of action to change read / write capabilities on another database server. If you require a fail-safe replication system, you should select an update-anywhere replication system.

# Update-Anywhere Replication System

An update-anywhere replication system allows peer-to-peer update capabilities. This capability allows users to function autonomously even when other systems or networks in the replication system are not available.

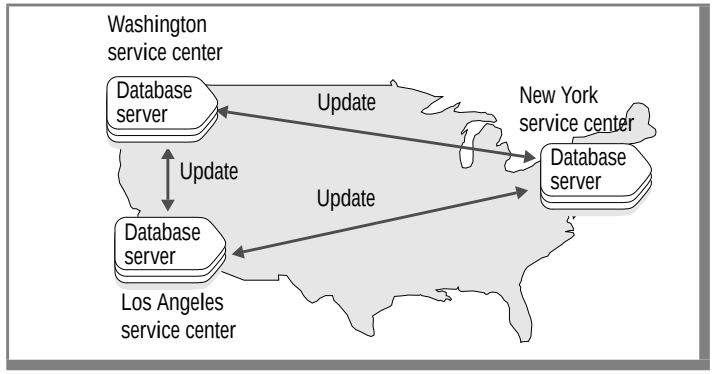



***Figure 4-5***
*Update-Anywhere Replication System*

Figure 4-1 illustrates an update-anywhere replication system. Because each service center can update a copy of the data, conflicts can occur when the data is replicated to the other sites. To resolve update conflicts, you must provide conflict detection and conflict resolution rules.

## Facts to Consider

Review the following information before you select your update-anywhere replication system.

### *Selecting Data for Replication*

The requirements for unique primary keys limit the use of serial keys in replication environments. A serial-column key cannot be a primary key by itself. The primary key must include at least one additional column besides the serial column. The additional column must make the combination of the additional column and the serial column globally unique across all database servers that participate in the replication system. For example, the additional column might be the location of the database server.

When Enterprise Replication applies rows that contain a serial-data column at a target database server, the serial-column value is set to the value assigned at the source database server.

### Administering the Replication System

Update-anywhere replication systems allow peer-to-peer updates, and therefore require conflict-detection or conflict-resolution rules. Administration of this type of replication system requires more administrative steps than the previously mentioned replication systems.

### Delivering Consistent Information

Some risk is associated with delivering consistent information in an update-anywhere replication system. You determine the amount of risk based on the type of conflict-resolution rules and procedures you choose with which to resolve conflicts. You can configure an update-anywhere replication system where no data is ever lost, however, you might find that other factors (for example, performance) outweigh your need for a fail-safe mechanism to deliver consistent information.

### Planning for Capacity

All replication systems require you to plan for capacity changes. For more information, see Chapter 5, "Setting Up Your Replication Environment."

### Planning for High Availability

High availability is a key element in the update-anywhere replication system. Because all database servers have read / write capabilities, you can easily remove or add database servers from the replication system. Select an update-anywhere replication system if your business requires a fail-safe system.

Chapter

# 5

# Setting Up Your Replication Environment


In This Chapter . . . . . 5-3

Operational Limitations . . . . . 5-3

SQL Statement Use . . . . . 5-3
- Forbidden SQL Statements . . . . . 5-4
- Limited SQL Statements . . . . . 5-4
- Permitted SQL Statements . . . . . 5-4

Capacity Planning . . . . . 5-5
- Logical-Log Records . . . . . 5-5
- Send and Receive Message Queues . . . . . 5-6
- Delete Tables . . . . . 5-7
- Spooling Directories . . . . . 5-7

Configuration Parameters . . . . . 5-8

Enterprise Replication Threads . . . . . 5-8

Environmental Considerations . . . . . 5-9
- Time Synchronization . . . . . 5-9
- Transaction-Processing Effect . . . . . 5-10
  - Using Unbuffered Logging . . . . . 5-10
  - Analyzing Replication Volume . . . . . 5-10
  - Using Complex SELECT Statements . . . . . 5-10

Enterprise Replication Data Types . . . . . 5-11
- Replicating on Heterogeneous Hardware . . . . . 5-11
- Replicating Simple Large Objects . . . . . 5-11
  - Replicating from Tablespaces or Blobspaces . . . . . 5-11
  - Conflict Resolution for Simple Large Objects . . . . . 5-12
  - Evaluating BYTE and TEXT Data with SPL Procedures . . . . 5-13

Replicating User-Defined Data Types . . . . . 5-15
Replicating Smart Large Objects . . . . . 5-15

Using GLS with Enterprise Replication . . . . . 5-15

## In This Chapter

Before you install Enterprise Replication, you need to determine what changes to make to your open-systems environment. This chapter discusses the various operational limitations, capacity requirements, and system changes you need to make based on the Enterprise Replication system you select. The chapter also discusses the data types that you can use for replication.

## Operational Limitations

Enterprise Replication imposes the following operational limits:

- Enterprise Replication supports replication only on Dynamic Server.
- High-availability data replication (HDR) cannot run concurrently with Enterprise Replication.
- Replication is restricted to base views. That is, you cannot define a replicate on a table that is a view.
- A database server can participate only once in a replicate.

## SQL Statement Use

After you define Enterprise Replication on a table by including that table as a participant in a replicate, you cannot execute any operation that changes the structure of the table.

## Forbidden SQL Statements

You cannot use the following SQL statements against a table that is included in a replicate:

- DROP TABLE
- RENAME TABLE
- ALTER FRAGMENT

## Limited SQL Statements

The following additional limitations also apply to tables defined for replication:

- Do not remove the primary key constraint.
- Do not rename, add, or drop columns.
- Do not modify a column type.
- Do not add or drop rowids.
- Do not add or drop CRCOLS.
- Do not modify the primary-key column(s).

  For example, do not alter the column to add default values or other constraints.
- Do not create clustered indexes.

## Permitted SQL Statements

Enterprise Replication permits the following SQL statements with no limitations:

- SET object mode
- START VIOLATIONS TABLE
- STOP VIOLATIONS TABLE
- CREATE TRIGGER
- DROP TRIGGER
- CREATE VIEW

- DROP VIEW
- CREATE SYNONYM
- DROP SYNONYM
- CREATE PROCEDURE
- DROP PROCEDURE
- ADD INDEX
- DROP INDEX
- ALTER INDEX (except TO CLUSTER)
- ALTER TABLE constraints (except for the primary key)

***Important:*** *Do not create a clustered index on a table after you have defined a replicate on that table; replication on the table will cease.*

# Capacity Planning

The following guidelines are provided to help you plan for the additional capacity Enterprise Replication uses.

## Logical-Log Records

The database server uses the logical log to store a record of changes to the data since the last archive. Enterprise Replication requires the logical log to contain entire row images for updated rows, including deleted rows.

The database server normally logs only columns that have changed. This behavior is called the Logical-Log Record reduction option. Enterprise Replication deactivates this option for tables that participate in Enterprise Replication. (The Logical-Log Record reduction option remains enabled for tables that do *not* participate in Enterprise Replication.) Enterprise Replication logs all columns, instead of the columns that have changed, which consequently increases the size of your logical log.

When you determine the size of your logical log, examine your data activity for normal operations and for the replication system you defined. As discussed in the previous paragraph, defining replication on a table might cause your transactions to log more data.

When the database server switches from one logical log to the next, a processing burden is placed on Enterprise Replication. Avoid logical-log configurations that result in frequent switching from one logical log to the next. One logical log switch per hour during normal processing is a good target. Plan to have sufficient logical-log space to contain 12 to 24 hours of normal log activity.

## Send and Receive Message Queues

The term *message queue* refers to both the send queue and the receive queue. The send queue resides at the source database server and contains committed transaction information. The receive queue resides at the target database server and contains information received by the target database server for application at the table level.

The send and receive queues are dbspaces. To determine the size of your send queue, determine your log usage and then consider how much data you can collect if your network goes down. For example, assume that you usually log 40 megabytes of data each day, but only 10 percent of that is replicated data. If your network is down for 24 hours, and you estimate that you will log 4 megabytes of replicated data each day, the dbspace you identify for the send queue must be at least 4 megabytes.

Receive queues are stably stored only when Enterprise Replication receives very large transactions. The receive queues usually do not require as much space as the send queue.

The send- and receive-queue dbspaces default to the root dbspace. You cannot change the send- and receive-queue dbspaces after you define replication for the database server.

For an active replication environment, you should create dbspaces for the send and receive queues before you define the replication server and assign the queues to those spaces. If the root dbspace fills, the Enterprise Replication and other database servers become unavailable.

The CDR_QUEUEMEM configuration parameter affects the amount of memory space that is available for the message queues. For information about CDR_QUEUEMEM, refer to Appendix A, “Configuration Parameters.”

## Delete Tables

Enterprise Replication creates delete tables only if you use time stamp and / or a procedure written in SPL (Stored Procedure Language) as your conflict-resolution rules. *Delete tables* keep track of modified rows and are used for conflict resolution. Enterprise Resolution creates and manages these tables. Neither the end user nor the database administrator interacts directly with the delete tables.

Delete tables are created on the database server where the data originates and on all the database servers to which data gets replicated. Delete tables are stored in the same dbspaces as their base tables.

If the base table has 100 megabytes of data, but only half the rows might be deleted or modified, then 50 megabytes is a reasonable estimate for the size of the delete table.

***Tip:*** *Delete tables are automatically deleted when the last replicate defined with conflict resolution on a table is deleted.*

## Spooling Directories

The aborted-transactions spooling (ATS) and row-information spooling (RIS) hold information about failed transactions. The default location for these directories is **/tmp** (UNIX) or **\tmp** (Windows NT). Informix recommends that you not leave the spooling directories in the default location but assign them to directories that you control.

For information about ATS and RIS, refer to Chapter 10, "Diagnosing Enterprise Replication."

## Configuration Parameters

The database server configuration file ($ONCONFIG) includes the following configuration parameters that affect the behavior of Enterprise Replication:

- CDR_DSLOCKWAIT
- CDR_EVALTHREADS
- CDR_LOGDELTA
- CDR_NIFCOMPRESS
- CDR_NIFRETRY
- CDR_NUMCONNECT
- CDR_QUEUEMEM

These parameters are documented in Appendix A, "Configuration Parameters." For additional information about configuration parameters, refer to your *Administrator's Guide*.

## Enterprise Replication Threads

The following table summarizes the threads that Enterprise Replication uses. You can use this information about threads when you evaluate memory use.

| Number of Threads | Thread Description |
|---|---|
| 1 | Performs physical I/O from logical log. |
| 1 | Verifies potential replication and sends applicable log-record entries to Enterprise Replication fan-out thread. |
| 1 | Receives log entries and passes entries to evaluator thread. |
| n | Evaluates log entry to determine if it should be replicated. (*n* is the number of evaluator threads specified by CDR_EVALTHREADS.) This thread also performs transaction compression on the receipt of COMMIT WORK and sends completed replication messages to the queue process. |

(1 of 2)

| Number of Threads | Thread Description |
|---|---|
| 1 | Parses all WHERE clauses for replicate definitions. |
| 1 | Keeps track of current buffers in use. |
| 1 per connection | Sending thread for site. |
| 1 per connection | Receiving thread for site. |
| 2...n | Accepts acknowledgments from site. At least 2, up to a maximum of the number of active connections. |
| 2..n | Read TEXT and BYTE data from spooled receive queue. At least 2, up to a maximum of the number of active connections. |
| # CPUs... | This thread is created for parallel processing. The thread synchronizes data, receives the replication message and applies it to a local instance. The thread also performs conflict resolution and inserts information into delete tables if required.<br>At least one thread is created for each CPU virtual processor (VP). The maximum number of threads is 2*(number of CPU VPs). |
| 1 | Schedules internal Enterprise Replication events. |

(2 of 2)

# Environmental Considerations

You need to evaluate various factors for your environment. Each replication system that you create affects your environment.

## Time Synchronization

Any time you use replication that requires conflict resolution, the operating-system times of the database servers that participate in the replicate must be synchronized. For tools for synchronizing the times, refer to the manuals for your particular operating system.

## Transaction-Processing Effect

Many variables affect the impact that replicating data has on your transaction processing. This section discusses some of these variables.

### Using Unbuffered Logging

Informix recommends that you replicate tables only from databases created with unbuffered logging.

Enterprise Replication evaluates the logical log for transactions that modify tables defined for replication. If a table defined for replication resides in a database that uses buffered logging, the transactions are not immediately written to the logical log, but are instead buffered and then written to the logical log in a block of logical records. When this occurs, Enterprise Replication evaluates the buffer of logical-log records all at once, which consumes excess CPU time and memory. When you define a table for replication in a database created with unbuffered logging, Enterprise Replication can evaluate the transactions as they are produced.

### Analyzing Replication Volume

Determine how many data rows change per day. For example, an application issues a simple INSERT statement that inserts 100 rows. If this table is replicated, these 100 rows must be propagated and analyzed before they are applied to the targets.

### Using Complex SELECT Statements

Evaluate the complexity of the SELECT statement that specifies the data for replication. If you define your replication system with a SELECT *.*, Enterprise Replication is optimized and immediately continues to process. If your SELECT statement includes a WHERE clause, Enterprise Replication must evaluate the WHERE clause, which can affect performance.

# Enterprise Replication Data Types

Enterprise Replication supports all Informix-defined data types. It also provides limited support for user-defined data types. Enterprise Replication does not support optical devices.

If you use SERIAL or SERIAL8 data types, you must define the serial columns *only* on the primary server. On the secondary server(s), the columns must be integers.

The following sections describe how Enterprise Replication captures, evaluates, and distributes simple large objects, smart large objects, and user-defined data types. For general information on data types, refer to the *Informix Guide to SQL: Reference* and *Informix Guide to Database Design and Implementation*.

## Replicating on Heterogeneous Hardware

Enterprise Replication supports all primitive data types across heterogeneous hardware. If you define a replicate that includes non-primitive data types (for example, BYTE and TEXT data), then the application must resolve data-representation issues that are architecture dependent.

If you use floating-point data types with heterogeneous hardware, you might need to use canonical format for the data transfers. Canonical message format sends floating-point values in a machine-independent format.

## Replicating Simple Large Objects

BYTE and TEXT data types are simple large objects. These data types can be stored in the tablespace with the rest of the table columns or in blobspaces.

### *Replicating from Tablespaces or Blobspaces*

BYTE and TEXT data that is stored in tablespaces (rather than in a blobspace) is placed in the logical log. Enterprise Replication reads the logical log to capture and evaluate BYTE and TEXT data for potential replication.

BYTE and TEXT data that is stored in blobspaces is *not* placed in the logical log. Enterprise Replication retrieves BYTE and TEXT data from blobspaces before sending the data to the target database server. However, if the replication operation is a delete, Enterprise Replication does not send BYTE or TEXT data to the target database server.

BYTE and TEXT data that is stored in blobspaces is not recorded in the logical log, but the transaction itself is logged. Such transactions can be replicated, but the BYTE or TEXT data is retrieved from the blobspace itself at the time that Enterprise Replication is preparing to send the transaction to the target system. Thus it is possible that the BYTE or TEXT data has been modified subsequent to the transaction currently being replicated. If Enterprise Replication encounters BYTE or TEXT data that has been modified since the transaction currently being replicated, it sends the data that it can successfully retrieve and flags the non-retrievable BYTE or TEXT data as undeliverable. However, in most cases, the action (update) that caused the BYTE or TEXT data to be modified is waiting to be delivered, so the data again becomes consistent. The data is inconsistent only in a small window of time.

### Conflict Resolution for Simple Large Objects

The following information explains how Enterprise Replication evaluates conflicts in an update-anywhere replicate with a time-stamp conflict-resolution rule and a row-by-row conflict-resolution scope.

When a replicated BYTE or TEXT column is modified on the source database server, Enterprise Replication records **cdrserver** and **cdrtime** for that column. (For more information on **cdrserver** and **cdrtime**, see "Shadow Columns" on page 6-4.) If the BYTE or TEXT column on the target database server is also stored in a blobspace, Enterprise Replication evaluates the **cdrserver** and **cdrtime** values in the source and target columns and uses the following logic to determine if the data is to be applied:

1. If the column of the replicated data has a time stamp that is greater than the time stamp of the column on the local row, the data for the column is accepted for replication.
2. If the server ID and time stamp of the replicated column are equal to the server ID and time stamp on the column on the local row, the data for the column is accepted for replication.

If either condition is true, the data for the column is accepted for replication even if the rest of the data (that is neither BYTE nor TEXT) is rejected. This can occur in certain situations where transactions arrive out-of-order.

### Evaluating BYTE and TEXT Data with SPL Procedures

If the replicate is defined with a stored-procedure conflict-resolution rule, the procedure must return the desired action for each BYTE or TEXT column. When the procedure is invoked, information about each BYTE or TEXT column is passed to the procedure as five separate fields. The following table describes the fields.

| Argument | Description |
|---|---|
| Column size (INTEGER) | NULL if the column is NULL.<br>The size of the column (if data exists for this column). |
| Blob flag [CHAR(1)] | For the local row, the field is always NULL.<br>For the replicated row:<br>■ D - BYTE or TEXT data is sent from the source database server.<br>■ U - BYTE or TEXT data is unchanged on the source database server. |
| Column type [CHAR(1)] | P indicates tablespace data.<br>B indicates blobspace data. |
| ID of last update server [CHAR(18)] | The ID of the database server that last updated this column for tablespace data.<br>Null for the blobspace data. |
| Last update time (DATETIME YEAR TO SECOND) | For the tablespace data: The date and time when the data was last updated.<br>For blobspace data: NULL. |

If the procedure returns an action code of A, D, I, or U, the procedure parses the return values of the replicated columns. Each BYTE or TEXT column can return a two-character field. For information about the action codes, refer to "Stored-Procedure Conflict-Resolution Rule" on page 7-17.

The first character defines the desired option for the BYTE or TEXT column, as the following table shows.

| Value | Function |
|---|---|
| C | Performs a time-stamp check for this column as used by the time-stamp rule. |
| N | Sets the replicate column to null. |
| R | Accepts the replicated data as it is received. |
| L | Retains the local data. |

The second character defines the desired option for blobspace data if the data is found to be undeliverable, as the following table shows.

| Value | Function |
|---|---|
| N | Sets the replicated column to null. |
| L | Retains the local data. This is the default. |
| O | Aborts the row. |
| X | Aborts the transaction. |

## Distributing BYTE and TEXT Data

If Enterprise Replication processes a row and discovers undeliverable BYTE or TEXT columns, the following actions can occur:

- Set any undeliverable columns to null if the replication operation is an INSERT and the row does not already exist at the target.
- Retain the old value of the local row if the replication operation is an UPDATE or if the row already exists on the target.

IDS

## Replicating User-Defined Data Types

Enterprise Replication supports all built-in Informix data types across heterogeneous platforms. The following restrictions apply to user-defined types (UDTs) in the SELECT statement of the participant modifier:

- The primary key cannot be a UDT.
- The replication must be between homogeneous platforms.
- The WHERE clause of the participant modifier cannot contain user-defined types.
- The data for multirepresentational UDTs (complex and collection types) must be stored in the row. Because the user has no control over when such data moves from the row into a smart large object, Informix recommends that you avoid multirepresentational UDTs.

## Replicating Smart Large Objects

This version of Enterprise Replication does not support replication of smart large objects (BLOBs and CLOBs).

GLS

# Using GLS with Enterprise Replication

An Enterprise Replication system can include databases in different locales, with the following restrictions:

- When you define a database server for Enterprise Replication, that server must be running in the English locale.

  In other words, the **syscdr** database on every Enterprise Replication server must be in the English locale.
- You can replicate only between databases that are in the same locale.

- Replication names can be in the locale of the database.
- Replication names might be displayed incorrectly by the Replication Manager.

  The Replication Manager can run in any locale, but it can run in only one locale at a time. If the databases in your replication system are in multiple locales, the names of some of the replicates might display incorrectly.

Codeset conversion with the GLS library requires only those codeset conversion files found in the **$INFORMIXDIR/gls/cv9** directory. In addition, ERM uses the UTF8 conversion and locale files.

7.3

For users of U.S. English, locales should be taken care of automatically by the database server installation and IECC setup. ♦

9.2

For users of U.S. English, locales should be taken care of automatically by the Client SDK / Informix Connnect installation and setup. ♦

If you use a non-U.S. English locale, you might need to explicitly provide the locale and conversion files.

For more information about how to use GLS, refer to *Informix Guide to GLS Functionality*.

Chapter

# 6

# Preparing Data for Replication


In This Chapter . . . . . 6-3

Preparing Consistent Data . . . . . 6-3
- Shadow Columns . . . . . 6-4
- Using the High-Performance Loader . . . . . 6-4
- Using the onunload and onload Utilities . . . . . 6-5
- Using the dbexport and dbimport Utilities . . . . . 6-5
- Using the UNLOAD and LOAD statements . . . . . 6-5

Blocking Replication . . . . . 6-6

Data Preparation Examples . . . . . 6-8
- Adding a New Participant to an Existing Replicate . . . . . 6-8
- Beginning Work Without Replication . . . . . 6-9
  - Using DB-Access to Begin Work Without Replication . . . . 6-10
  - Using ESQL/C to Begin Work Without Replication . . . . . 6-10

## In This Chapter

The goal of data replication is to provide identical, or at least consistent, data on multiple database servers. This chapter describes how to prepare the information in your databases for replication.

When you define a new replicate on tables with existing data on different database servers, the data might not be consistent. Similarly, if you add a participant to an existing replicate you need to ensure that all the databases in the replicate have consistent values.

You might also need to take special steps when you recover data. For example, if a dbspace fills up on a target database server, you can recover the data from the ATS files. However, it might be easier to unload the data from the source and apply that data to the target database.

## Preparing Consistent Data

In most cases, preparing consistent data simply requires that you decide which of your databases has the most accurate data and then copy that data onto the target database. If the target database already has data, for data consistency you must remove that data before adding the copied data.

For copying data, you can use the following tools:

- The High-Performance Loader (HPL)
- The **onunload** and **onload** utilities
- The **dbexport** and **dbimport** utilities
- The UNLOAD and LOAD statements

When you unload and load data, you must use the same type of utility for both the unload and load operations. For example, you cannot unload data with the **onunload** utility and then load the data with a LOAD statement.

## Shadow Columns

In an update-anywhere replication, you must provide for conflict resolution. When you create a table that uses the time stamp or time stamp plus stored-procedure conflict-resolution rule, you must define shadow columns with the WITH CRCOLS clause. The shadow columns, **cdrserver** and **cdrtime**, contain information that Enterprise Replication uses for conflict resolution. If a table does not have shadow columns, you must use the ignore conflict-resolution rule. For more information about the WITH CRCOLS clause, refer to the *Informix Guide to SQL: Syntax*.

If a table that you plan to replicate includes **cdrserver** and **cdrtime**, the statements that you use for unloading the data must explicitly name the shadow columns. If you use * FROM *tablename* to the data to unload, the data from the shadow columns will not be unloaded. To include the shadow columns in the unloaded data, use a statement like the following statement:

```
SELECT *, cdrserver, cdrtime FROM tablename
```

## Using the High-Performance Loader

The HPL provides a high-speed tool for moving data between databases. For information about HPL, refer to the *Guide to the High-Performance Loader*.

How you use the HPL depends on how you defined the tables to replicate. If the table definition included the WITH CRCOLS clause, then you must take special steps when you unload the data.

When you use the WITH CRCOLS clause with the CREATE TABLE statement, the database server creates two shadow columns, **cdrserver** and **cdrtime**, that contain replication information.

If the table contains shadow columns, you must perform the following additional steps:

1. Include the **cdrserver** and **cdrtime** columns explicitly in the query statement when you unload the data.
2. Include the **cdrserver** and **cdrtime** columns in your map when you load the data.
3. Use Express mode to load data that contains the **cdrserver** and **cdrtime** columns. You must perform a level-0 archive after completion.

You can also use deluxe mode without replication to load data. After a deluxe mode load, you do not need to do a level-0 archive. Deluxe mode also allows you to load TEXT and BYTE data and user-defined types.

## Using the onunload and onload Utilities

You can use the **onunload** and **onload** utilities only to unload and load an entire table. If you want to unload selected columns of a table, you must use either **unload** or the HPL.

For more information about **onunload** and **onload**, see the *Informix Migration Guide.*

## Using the dbexport and dbimport Utilities

If you need to copy an entire database for replication, you can use the **dbexport** and **dbimport** utilities. These utilities unload an entire database, including its schema, and then re-create the database. If you want to move selected tables or selected columns of a table, you must use some other utility.

For more information about **dbexport** and **dbimport**, see the *Informix Migration Guide.*

# Using the UNLOAD and LOAD statements

The UNLOAD and LOAD statements allow you to move data within the context of an SQL program.

If the table contains shadow columns, you must perform the following two additional steps:

1. Include the **cdrserver** and **cdrtime** columns explicitly in your statement when you unload your data.
2. List the columns that you want to load in the INSERT statement and explicitly include the **cdrserver** and **cdrtime** columns in the list when you load your data.

For more information about the UNLOAD and LOAD statements, see the *Informix Guide to SQL: Syntax.*

## Blocking Replication

If the table that you are preparing for replication is in a database that already uses replication, you might need to block replication while you prepare the table. The BEGIN WORK WITHOUT REPLICATION statement starts a transaction that does not replicate to other database servers.

The following code fragment shows how you might use this statement:

```
BEGIN WORK WITHOUT REPLICATION
LOCK TABLE office
DELETE FROM office WHERE description = 'portlandR_D'
COMMIT WORK
```

The following list indicates actions that occur when a transaction starts with BEGIN WORK WITHOUT REPLICATION:

- All constraint checking for the transaction is automatically set to deferred.
- SQL does not generate any values for the shadow columns (**cdrserver** and **cdrtime)** for the rows that are inserted or updated within the transaction. You must supply values for these columns with the explicit column list. You must supply values for the shadow columns even if you want the column values to be NULL.

- If you use the WITH CRCOLS clause to modify a table that is already defined in Enterprise Replication, you must explicitly list the columns to be modified. The following two examples show an SQL statement and the correct changes to the statement to modify columns:
  - If **tablename1** is a table defined for replication, the statement:

    ```
    LOAD FROM filename INSERT INTO tablename1;
    ```

    must be changed to:

    ```
    LOAD FROM filename INSERT INTO tablename1 (list of
    columns);
    ```

    The list of columns must match the order and the number of fields in the load file.
  - If **tabname1** and **tabname2** are tables defined for replication with the same schema, the statement:

    ```
    INSERT INTO tabname1  * FROM tabname2;
    ```

    must be changed to an explicit statement, where *col1*, *col2*,..., *colN* are the columns of the table:

    ```
    INSERT INTO tabname1 VALUES
           (cdrserver, cdrtime, col1, col2,..., colN)
        cdrserver, cdrtime, *
       FROM tabname2;
    ```

    The shadow columns (**cdrserver**, **cdrtime**) are not included in an * list.

For more information about these statements, refer to the *Informix Guide to SQL: Syntax*.

# Data Preparation Examples

The following examples illustrate how to prepare data for replication.

## Adding a New Participant to an Existing Replicate

The following steps provide an example of how to add a new participant (**delta**) to an existing replicate using the LOAD, UNLOAD, and BEGIN WORK WITHOUT REPLICATION statements. Replicate **zebra** replicates data from table **table1** for the following database servers: **alpha**, **beta**, and **gamma**.

The servers **alpha**, **beta**, and **gamma** belong to the server groups **g_alpha**, **g_beta**, and **g_gamma**, respectively. Assume that **alpha** is the database server from which you want to get the initial copy of the data.

1. Declare server **delta** to Enterprise Replication. For example,

   ```
   cdr def ser -c delta -I -S g_alpha g_delta
   ```

   At the end of this step, all servers in the replication environment include information about **delta**, and **delta** has information about all other servers.

2. Add **delta** as a participant to replicate **zebra**. For example,

   ```
   cdr cha rep -a zebra "dbspace@g_delta:owner.table1"
   ```

   This step updates the replication catalog. The servers **alpha**, **beta**, and **gamma** do not queue any qualifying replication data for **delta** because the replicate on **delta**, although defined, has not been started.

3. Suspend server to **delta** on **alpha**, **beta**, and **gamma**.

   ```
   cdr sus ser g_alpha g_beta g_gamma
   ```

   As a result of this step, replication data is queued for **delta**, but no data is delivered.

4. Start replication for replicate **zebra** on **delta**.

   ```
   cdr sta rep zebra g_delta
   ```

   This step causes servers **alpha**, **beta**, and **gamma** to start queueing data for **delta**. No data is delivered to **delta** because **delta** is suspended. **delta** queues and delivers qualifying data (if any) to the other servers.

   Informix recommends that you do no transactions on **delta** that affect table **table1** until you finish the synchronization process.

5. Unload data from table **table1** using the UNLOAD statement or the **unload** utility on HPL.

6. Copy the unloaded data to **delta**.

7. Start transactions with BEGIN WORK WITHOUT REPLICATION, load the data using the LOAD statement, and commit the transaction.

   If you used the HPL to unload the data in step 5, then use the HPL Deluxe load without replication to load the data into **table1** on **delta**.

8. Resume server **delta** on **alpha**, **beta**, and **gamma**.

   This step starts the flow of data from **alpha**, **beta**, and **gamma** to **delta**.

   At this point you might see some transactions aborted because of conflict. Transaction can abort because **alpha**, **beta**, and **gamma** started queueing data from **delta** in step 4. However, those same transactions might have been moved in steps 5 and 7.

It might seem strange to declare replication on server **delta** and then immediately suspend replication. You must do this because, while you are preparing the replicates and unloading and loading files, the other servers in the replicate (**alpha**, **beta**, and **gamma**) might be collecting information that needs to be replicated. After you finish loading the initial data to **delta** and resume replication, the information that was generated during the loading process can be replicated.

## Beginning Work Without Replication

You might need to put data into a database that you do not want replicated, perhaps for a new server or because you had to drop and re-create the table.

### Using DB-Access to Begin Work Without Replication

The following example shows how to use DB-Access to begin work without replication as well as update the Enterprise Replication shadow columns **cdrserver** and **cdrtime**:

```
DATABASE adatabase;
BEGIN WORK WITHOUT REPLICATION
INSERT into mytable (cdrserver, cdrtime, col1, col2, ....)
    VALUES (10, 845484154, value1, value2, ....);
UPDATE mytable
    SET cdrserver = 10, cdrtime = 845484154
    WHERE col1 > col2;
COMMIT WORK
```

### Using ESQL/C to Begin Work Without Replication

The following example shows how to use ESQL/C to begin work without replication as well as update the Enterprise Replication shadow columns **cdrserver** and **cdrtime**:

```
MAIN (argc, argv)
    INT argc;
    CHAR*argv[];
{
    EXEC SQL CHARstmt[256];
    EXEC SQL database mydatabase;

    sprintf(stmt, "BEGIN WORK WITHOUT REPLICATION");
    EXEC SQL execute immediate :stmt;

    EXEC SQL insert into mytable (cdrserver, cdrtime,
    col1, col2, ...)
    values (10, 845494154, value1, value2, ...);

    EXEC SQL update mytable
    set cdrserver = 10, cdrtime = 845494154
    where col1 > col2;
    EXEC SQL commit work;
}
```

***Important:*** *You must use the following syntax when you issue the BEGIN WORK WITHOUT REPLICATION statement from ESQL/C programs. Do not use the '$' syntax.*

```
sprintf(stmt, "BEGIN WORK WITHOUT REPLICATION");
EXEC SQL execute immediate :stmt;
```

# Replicating Data

Section III


Chapter 7 Enterprise Replication Processes

Chapter 8 Enterprise Replication Menus

Chapter 9 Enterprise Replication Manager Wizards

Chapter 10 Diagnosing Enterprise Replication

Chapter 11 Command-Line Utility

Chapter 12 API Functions

Chapter

# 7

# Enterprise Replication Processes


In This Chapter . . . . . 7-3

Capturing Transactions . . . . . 7-3

Evaluating Data for Replication . . . . . 7-4
- Evaluating Time . . . . . 7-4
- Evaluating Images of a Row . . . . . 7-5
- Evaluating Replication Examples . . . . . 7-7
- Evaluating Primary-Key Updates . . . . . 7-10
- Receiving and Sending Data from the Message Queue . . . . . 7-11

Applying Data for Distribution . . . . . 7-12
- Choosing Conflict-Resolution Rules and Scopes . . . . . 7-13
- Time-Stamp Conflict-Resolution Rule . . . . . 7-15
  - Stored-Procedure Conflict-Resolution Rule . . . . . 7-17
  - Ignore Conflict-Resolution Rule . . . . . 7-20

Using Cascading Deletes and Triggers . . . . . 7-20
- Using Cascading Deletes . . . . . 7-21
- Using Triggers . . . . . 7-21

## In This Chapter

Enterprise Replication uses various processes to gather and replicate data. This chapter discusses the processes that replicate data after you define your replication system:

- Capturing transactions
- Evaluating data for replication
- Distributing replication messages

## Capturing Transactions

As the replication server writes rows to the logical log, it marks rows that should be replicated. Later, Enterprise Replication reads the logical log to obtain the row images for tables that participate in a replicate.

Informix database servers manage the logical log in a circular fashion, where the most recent logical-log entries write over the oldest entries. If the database server comes close to overwriting a logical log that Enterprise Replication has not yet processed, user transactions are blocked until Enterprise Replication can advance. This situation, where user transactions are blocked, happens only if the system is severely misconfigured. Reading the logical log must advance far enough to prevent new logical-log entries from overwriting the logs Enterprise Replication has not yet processed.

The row images that participate in replicates are passed to Enterprise Replication for further evaluation.

# Evaluating Data for Replication

Enterprise Replication evaluates transactions based on a distinct time and a change to a final image of a row.

## Evaluating Time

Most Enterprise Replication commands are issued at a distinct time. A command does not completely take effect until Enterprise Replication processing passes the time in the logical log at which the command was issued.

Figure 7-1 illustrates how Enterprise Replication determines the correct data to capture and replicate based on a distinct time.

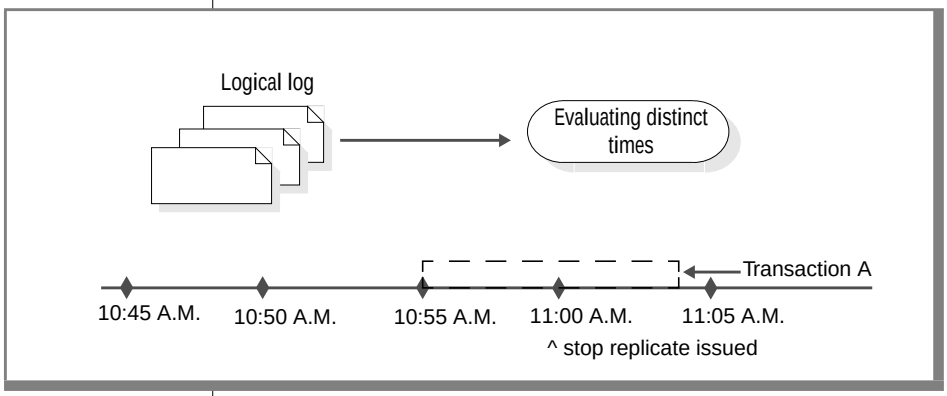



***Figure 7-1***
*Evaluating a Distinct Time*

At (the logical-log time) 10:45 A.M., Enterprise Replication begins to read logical-log records that are marked for replication. At 11:00 A.M., the user issues a **Stop Replicate** command to stop the replication of data in a replicate. Enterprise Replication refers to the time at which replication stops. Enterprise Replication evaluates all transactions that were committed between 10:45 A.M. and 11:00 A.M. and passes them to Enterprise Replication for evaluation.

Enterprise Replication does not replicate any transactions that are committed after 11:00 A.M. For example, Transaction A in Figure 7-1 is not passed to Enterprise Replication. Although the transaction starts at 10:55 A.M., it does not complete until 11:04 A.M., and therefore, does not commit before the cut-off time of 11:00 A.M.

## Evaluating Images of a Row

Enterprise Replication evaluates the initial and final images of a row and any changes that occur between the two row images to determine which rows to replicate. Each row image contains the data in the row as well as the action performed on that row.

Enterprise Replication evaluates the images of a row in parallel to assure high performance. Enterprise Replication collects records that qualify for replication into a transaction list and passes them to the Enterprise Replication *message queue* for distribution. Enterprise Replication uses two queues to send and receive messages: a send queue and a receive queue. The message queue refers to the send and receive queues.

A row might change more than once in a particular transaction. For example, a transaction might insert and then update a row prior to committing. Enterprise Replication evaluates the net effect (final state) of a transaction based on the row buffers in the log. Enterprise Replication then determines what should replicate based on the net effect, the initial state of the row, and whether the replicate definition (in particular, the WHERE clause) applies to the initial and final state.

The table in Figure 7-2 on page 7-6 shows the logic that determines which rows are candidates for replication.

***Figure 7-2***
*Enterprise Replication Evaluation Logic*

| Initial Image | Replicate Evaluates | Final Image | Replicate Evaluates | Primary-Key Changed? | Send to Destination Database Server | Comments |
|---|---|---|---|---|---|---|
| INSERT | T *or* F | DELETE | T *or* F | yes *or* no | nothing | Net change of transaction results in no replication |
| INSERT | T *or* F | UPDATE | T | yes *or* no | INSERT with final row image | Inserts final data of transaction |
| INSERT | T *or* F | UPDATE | F | yes *or* no | nothing | Final evaluation determines no replication |
| UPDATE | T | DELETE | T *or* F | yes *or* no | DELETE with initial row image | Net result of transaction is delete |
| UPDATE | F | DELETE | T *or* F | yes *or* no | nothing | Net change of transaction results in no replication |
| UPDATE | T | UPDATE | T | yes | DELETE with initial row image and INSERT with final row image | Ensures old primary key does not replicate |
| UPDATE | T | UPDATE | T | no | UPDATE with final row image | Simple update |
| UPDATE | T | UPDATE | F | yes *or* no | DELETE with initial row image | Row no longer matches replicate definition |
| UPDATE | F | UPDATE | T | yes *or* no | INSERT with final row image | Row now matches replicate definition |
| UPDATE | F | UPDATE | F | yes *or* no | nothing | No match exists and therefore, no replication |

The table in Figure 7-2 illustrates how Enterprise Replication evaluates the row-image type (INSERT, UPDATE, DELETE), the results of evaluating the replicate WHERE clause for both the initial and final image, and whether the primary key changes as a result of the transaction

***Tip:*** *The evaluation logic in Figure 7-2 on page 7-6 assumes that the source and the destination tables are synchronized. If the tables are not synchronized, anomalous behavior could result.*

## Evaluating Replication Examples

Figure 7-3, Figure 7-5 on page 7-8, and Figure 7-7 on page 7-9 show three examples of how Enterprise Replication uses logic to evaluate transactions for potential replication.

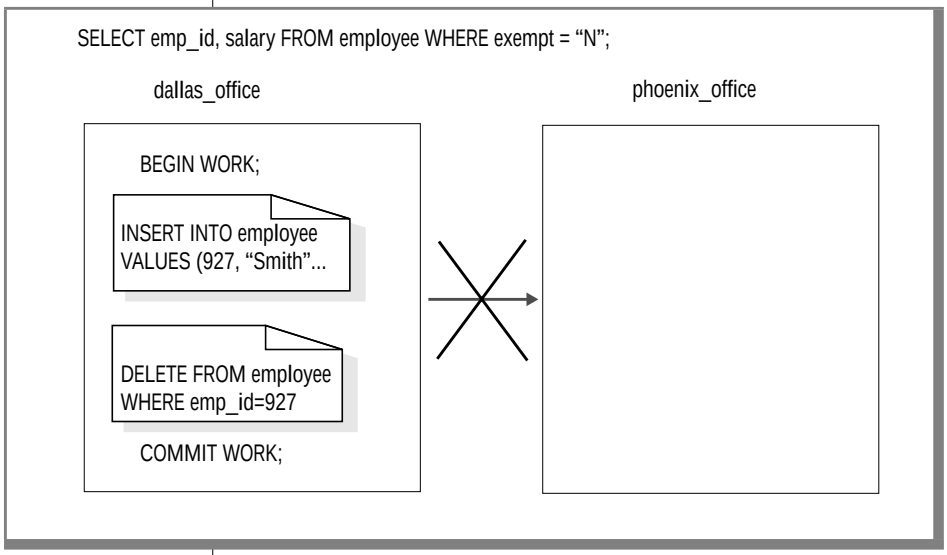



***Figure 7-3***
*Insert Followed by a Delete*

Figure 7-3 shows a transaction that takes place at the Dallas office. Enterprise Replication uses the logic in Figure 7-4 on page 7-8 to evaluate whether any information is sent to the destination database server at the Phoenix office.

**Figure 7-4**
*Insert Followed by a Delete Evaluation Logic*

| Initial Image | Replicate Evaluates | Final Image | Replicate Evaluates | Primary-Key Changed? | Send to Destination Database Server |
|---|---|---|---|---|---|
| INSERT | T *or* F | DELETE | T *or* F | yes *or* no | nothing |

Enterprise Replication determines that the insert followed by a delete results in no replication operation, therefore, a replication message is not sent.

In Figure 7-5, Enterprise Replication uses the logic in Figure 7-6 to evaluate whether any information is sent to the destination database server.

**Figure 7-5**
*Insert Followed by an Update*

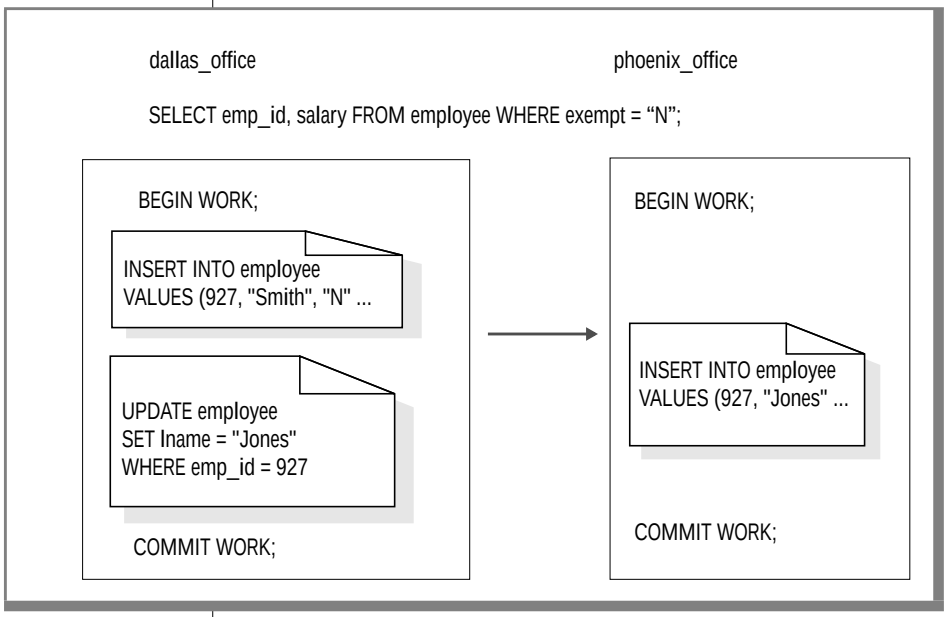


**Figure 7-6**
*Insert Followed by An Update Evaluation Logic*

| Initial Image | Replicate Evaluates | Final Image | Replicate Evaluates | Primary-Key Changed? | Send to Destination Database Server |
|---|---|---|---|---|---|
| INSERT | T *or* F | UPDATE | T | yes or no | INSERT with final row image |

The replicate WHERE clause imposes the restriction that only rows with the **exempt** column equal to "N" are replicated. Enterprise Replication evaluates the transaction (an insert followed by an update) and converts it to an insert to propagate the updated image (the final image).

In Figure 7-7, Enterprise Replication uses the logic in Figure 7-8 to evaluate whether any information is sent to the destination database server.

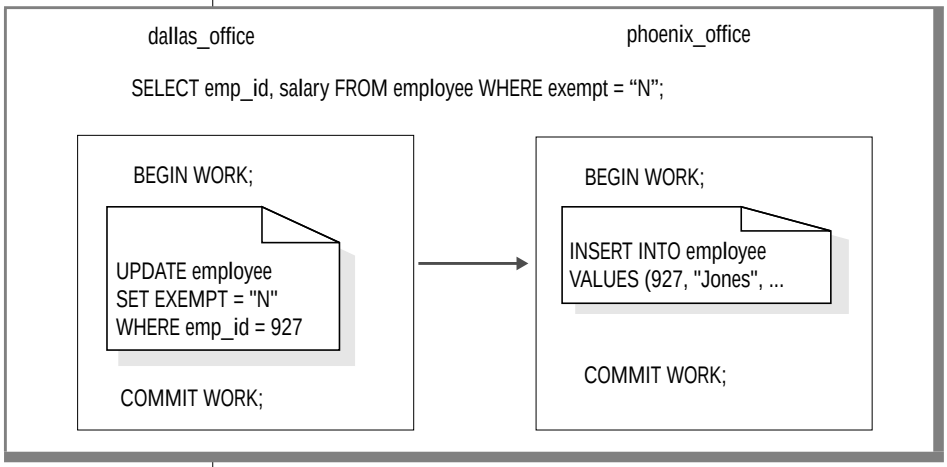



**Figure 7-7**
*Update; Not Selected to Selected*

**Figure 7-8**
*Update; Not Selected to Selected Evaluation Logic*

| Initial Image | Replicate Evaluates | Final Image | Replicate Evaluates | Primary-Key Changed? | Send to Destination Database Server |
|---|---|---|---|---|---|
| UPDATE | F | UPDATE | T | yes *or* no | INSERT with final row image |

The example shows a replicate WHERE clause column update. A row that does not meet the WHERE clause selection criteria is updated to meet the criteria. Enterprise Replication replicates the updated row image and converts the update to an insert.

## Evaluating Primary-Key Updates

Enterprise Replication evaluates rows for primary-key updates, for WHERE clause column updates and for multiple updates to the same row. The following list describes an occurrence in a transaction and the Enterprise Replication action:

- Primary-key updates

  Enterprise Replication translates an update of a primary key into a delete of the original row and an insert of the row image with the new primary key. If triggers are enabled on the target system, insert triggers are executed.

- Multiple-row images in a transaction

  Enterprise Replication compresses multiple-row images and only sends the net change to the target. Triggers execute only once.

- WHERE clause column updates

  If a replicate includes a WHERE clause in its data selection, the WHERE clause imposes selection criteria for rows in the replicated table.

  - If an update changes a row so that it no longer passes the selection criteria on the source, it is deleted from the target table. Enterprise Replication translates the update into a delete and sends it to the target.
  - If an update changes a row so that it passes the selection criteria on the source, it is inserted into the target table. Enterprise Replication translates the update into an insert and sends it to the target.

For a detailed example that illustrates the evaluation of data in a single transaction where one row is inserted and then updated several times, see "Evaluating Images in a Complex Transaction" on page D-1.

Enterprise Replication supports the replication of BYTE and TEXT data. Enterprise Replication implements the replication of BYTE and TEXT data somewhat differently from the replication of other data types. Read through these sections to understand the processes used to evaluate data for replication and then read "Replicating Simple Large Objects" on page 5-11.

## Receiving and Sending Data from the Message Queue

Enterprise Replication uses receive and send queues (also referred to as the message queues) to receive and deliver replication messages to and from database servers that participate in a replicate.

- Receive queue

  A dbspace that holds replication messages at the target database server until the target database server acknowledges receipt of the messages. The pathname that specifies the receive-queue dbspace can be no longer than 256 bytes. For more information, see Chapter 5, "Setting Up Your Replication Environment."

- Send queue

  A dbspace that stores replication messages to be delivered to target database servers that participate in a replicate. The pathname that specifies the send-queue bytes can be no longer than 256 characters.

Each message contains the filtered log records for a single transaction. Target sites acknowledge receipt of a message when the message is *safely stored*. A message is safely stored when it is in a stable (recoverable) queue or when it is in the target database.

If a target database server is unreachable, the replication messages remain in a stable queue at the source database server. Temporary failures are common and no immediate action is taken by the source database server; it continues to queue transactions. When the target database server becomes available again, queued transactions are transmitted and applied (see "Applying Data for Distribution" on page 7-12).

If the target database server is unavailable for an extended period, the send queue on the source database server might consume excessive resources. It might not be appropriate to save all transactions for the remote database server in this situation. To prevent unlimited transaction accumulation, the unavailable target database server can be removed from the replicate. The database server that is removed needs to synchronize with the other database servers before it rejoins any replicate. For more information, see Chapter 6, "Preparing Data for Replication."

## Applying Data for Distribution

Enterprise Replication applies the replicated data to target database servers.

When Enterprise Replication applies replication messages, it checks to make sure that no *collisions* exist. A collision occurs when potential simultaneous updates of the same data on different database servers are applied. Enterprise Replication reviews data one row at a time to detect a collision. Enterprise Replication scans to see if the same primary key already exists in the target table or in the associated *delete table*, or if a *replication order error* is detected. A delete table stores the row images of deleted rows. A replication order error is the result of a replication message that arrives from different database servers with one of the following illogical results:

- A replicated DELETE that finds no row to DELETE on the target
- An UPDATE that finds no row to UPDATE on the target
- An INSERT that finds a row that already exists on the target

If Enterprise Replication finds a collision, it must resolve the conflict before it applies the replication message to the target database server.

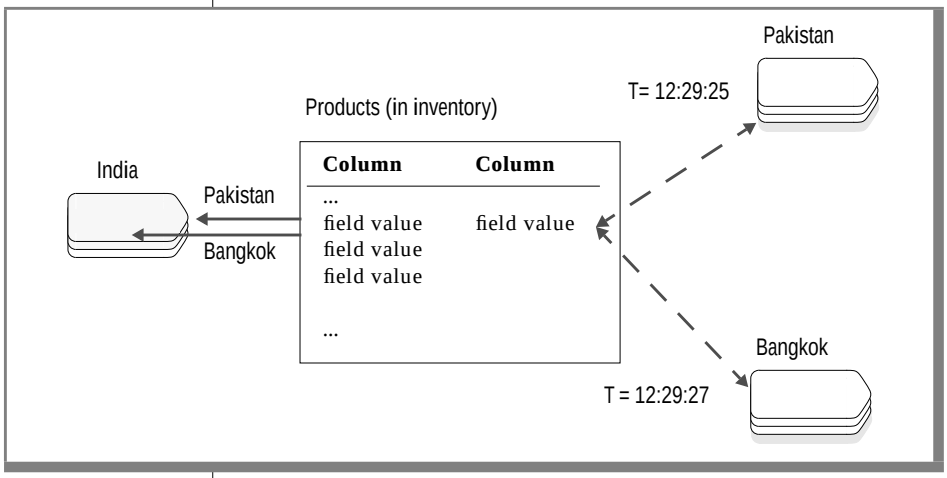



***Figure 7-9***
*Collision Example*

Figure 7-9 shows a situation that yields a conflict. Pakistan updates the row two seconds before Bangkok updates the same row. The Bangkok update arrives at the India site first, and the Pakistan update follows. The Pakistan time T is earlier than the Bangkok time. Because both updates involve the same data and a time discrepancy exists, Enterprise Replication detects a collision.

## Choosing Conflict-Resolution Rules and Scopes

You need to determine conflict-resolution rules and a conflict-resolution scope for each replicate.

***Tip:*** *ERM supports two operating modes, expert and nonexpert. Expert mode provides more options for control of your Enterprise Replication system than standard mode. The CLU (see Chapter 11) and the APIs (see Chapter 12) do not distinguish between these two modes. The CLU and APIs provide all expert-mode options.*

Enterprise Replication supports up to two conflict-resolution rules for each replicate. You can define a primary rule and a secondary rule (if desired). Figure 7-10 shows the valid combinations of Enterprise Replication conflict-resolution rules.

***Figure 7-10***
*Valid Conflict-Resolution Rule Combinations*

| Primary Rule | Secondary Rule | Enterprise Replication Operating-Mode Support |
|---|---|---|
| Time stamp | None | Expert |
| Time stamp | Stored procedure | Expert |
| Stored procedure | None | Expert |
| Ignore | None | Standard and expert |

Each conflict-resolution rule behaves differently depending on the conflict-resolution *scope* you chose. The following list describes each conflict-resolution scope:

- Row-by-row scope

  When you choose a row-by-row conflict-resolution scope, Enterprise Replication evaluates one row at a time. It only applies replicated rows that win the conflict resolution with the target row. If an entire replicated transaction receives row-by-row evaluation, some replicated rows are applied and other replicated rows might not be applied.

- Transaction scope

  When you choose a transaction conflict-resolution scope, Enterprise Replication applies the entire transaction if the replicated transaction wins the conflict resolution. If the target wins the conflict (or other database errors are present), the entire replicated transaction is not applied and rollback occurs.

  A transaction scope for conflict resolution guarantees transactional integrity.

***Important:*** *Enterprise Replication defers some constraint checking on the target tables until the transaction commits. If a unique constraint or foreign-key constraint violation is found on any row of the transaction at commit time, the entire transaction is rejected (regardless of the conflict-resolution scope).*

## Time-Stamp Conflict-Resolution Rule

The time-stamp rule evaluates the latest time stamp of the replication against the target and determines a winner. The time-stamp resolution rule behaves differently depending on which scope you choose for conflict resolution (row-by-row or transaction), as follows:

- Row-by-row conflict-resolution scope

  Enterprise Replication evaluates one row at a time. The row with the most recent time stamp wins the collision and is applied to the target database servers. If an SPL procedure is defined as a secondary conflict-resolution rule, the procedure resolves the conflict when the row times are equal.

- Transaction conflict-resolution scope

  Enterprise Replication evaluates the most recent row-update time among all the rows in the replicated transaction. This time is compared to the time stamp of the appropriate target row. If the time stamp of the replicated row is more recent than the target, the entire replicated transaction is applied. If a procedure is defined as a secondary conflict-resolution rule, it is used to resolve the conflict when the time stamps are equal.

***Tip:*** *A secondary procedure is invoked only if Enterprise Replication evaluates rows and discovers equal time stamps.*

If either conflict-resolution scope (row-by-row or transaction) evaluates rows and discovers equal time stamps, and no procedure is defined as a secondary conflict-resolution rule, the replicated row or transaction is applied.

Figure 7-11 on page 7-16 shows how a conflict is resolved based on the latest time stamp with row-by-row scope. The time stamp $T_{last\_update}$ represents the row on the target database server with the last (most recent) update. The time stamp $T_{repl}$ represents the time stamp on the incoming row.

Enterprise Replication first checks to see if a row with the same primary key exists in either the target table or its corresponding delete table.

If the row exists, Enterprise Replication uses the latest time stamp to resolve the conflict.

***Figure 7-11***
*Conflict Resolution Based on the Time Stamp*

| Row Exists on Target? | Time Stamp | Database Operation | | |
|---|---|---|---|---|
| | | **Insert** | **Update** | **Delete** |
| No | n/a | Apply row | Apply row<br>(Convert UPDATE to INSERT) | Apply row<br>(INSERT into Enterprise Replication delete table) |
| Yes | $T_{last\_update} < T_{repl}$ | Apply row<br>(Convert INSERT to UPDATE) | Apply row | Apply row |
| | $T_{last\_update} > T_{repl}$ | Discard row | | |
| | $T_{last\_update} = T_{repl}$ | Apply row if no procedure is defined as a secondary conflict-resolution rule. Otherwise, invoke the procedure. | | |

***Important:*** *Do not remove the delete tables created by Enterprise Replication. The delete table is automatically removed when the last replicate defined with conflict resolution is deleted.*

The time-stamp conflict-resolution rule assumes time synchronization between cooperating Enterprise Replication servers. All time stamps and internal computations are performed in Greenwich mean time and have an accuracy of plus or minus one second.

***Important:*** *Enterprise Replication does not manage clock synchronization between database servers that participate in a replicate.*

To synchronize the time on one database server with the time on another database server, use one of the following commands:

**UNIX**

```
rdate hostname
```

◆

**WIN NT**

```
net time \\servername /set
```

or

```
net time /domain:servername /set
```

♦

These commands do not guarantee the times will remain synchronized. Informix recommends that you use a product that supplies a network time protocol to ensure that times remain synchronized.

### Stored-Procedure Conflict-Resolution Rule

This rule allows you complete flexibility to determine which row prevails in the database. You can assign an SPL procedure as the primary conflict-resolution rule.

If you use an SPL procedure as a secondary conflict-resolution rule, the time-stamp conflict-resolution rule must be the primary rule.

Enterprise Replication passes the following information to an SPL procedure as arguments.

| Argument | Description |
|---|---|
| Server name [CHAR(18)] | From the local target row<br>(null if local target row does not exist) |
| Time stamp (DATETIME YEAR TO SECOND) | From the local target row<br>(null if local target row does not exist) |
| Local delete-table indicator [CHAR(1)]<br>or<br>Local key delete-row indicator [CHAR(1)] | ■ Y indicates the origin of the row is the delete table<br>■ K indicates the origin of the row is the replicate-key delete row (for a key update that is sent as a key delete and a key insert) because the local target row with the old key does not exist. |
| Server name [CHAR(18)] | Of the replicate source |
| Time stamp (DATETIME YEAR TO SECOND) | From the replicated row |
| Replicate action type [CHAR(1)] | I- insert<br>D- delete<br>U- delete update |
| Local row data returned in regular SQL format | Where the regular SQL format is taken from the SELECT clause of the participant list. |
| Replicate row data after-image returned in regular SQL format | Where the regular SQL format is taken from the SELECT clause of the participant list. |

The procedure must set the following arguments before the procedure can be applied to the replication message.

| Argument | Description |
|---|---|
| An indicator of the desired database operation to be performed [CHAR(1)] | Same as the replicate action codes with the following additional codes<br>■ A - accept the replicated row and resolve replication exception<br>■ S - accept the replicated row as received<br>■ O - discard the replicated row<br>■ X - abort the transaction |
| A non-zero integer value to request logging of the resolution and the integer value in the spooling file(s) (INTEGER) | Logging value takes effect only if logging is configured for this replicate. |
| The columns of the row to be applied to the target table replicate action type in regular SQL format | This list of column values is not parsed if the replicate action type that the procedure returns is S, O, or X. |

You can use the arguments to develop application-specific procedures. For example, you can create a procedure in which a database server always wins a conflict regardless of the time stamp. For an example, see "A Stored-Procedure Example" on page D-3. The following list includes some items to consider when you use a procedure for conflict resolution:

- Determining when the procedure executes.

  Optimized mode invokes the procedure if a collision is detected and the replicate source database server is either different from the server shadow column in the target row, or if the source update time is less than or equal to the target row.

  If you do not choose to optimize the execution of your procedures, a procedure executes *every* time Enterprise Replication detects a collision.
- Any action that a procedure takes as a *result* of replication does not replicate.
- Frequent use of procedures might affect performance.

***Tip:*** *Do not assign a procedure that is not optimized as a primary conflict-resolution rule for applications that predominantly insert rows successfully.*

### Ignore Conflict-Resolution Rule

The ignore conflict-resolution rule does not attempt to detect or resolve conflicts. A row or transaction applies successfully or it fails. A row might fail to replicate because of standard database reasons or replication order errors.

The ignore conflict-resolution rule can only be used as a primary conflict-resolution rule and can have either a transaction or a row-by-row conflict-resolution scope. Figure 7-12 illustrates the ignore conflict-resolution rule.

***Figure 7-12***
*Ignore Conflict-Resolution Rule*

| | Database Operation | | |
|---|---|---|---|
| **Row Exists in Target?** | **Insert** | **Update** | **Delete** |
| No | Apply row | Discard row | Discard row |
| Yes | Discard row | Apply row | Apply row |

When a replication message fails to apply on a target, you can spool the information to one or both of the following directories:

- Aborted-transaction spooling (ATS)

  If selected, all buffers in a failed replication message that compose a transaction are written to this directory.

- Row-information spooling (RIS)

  If selected, the replication message for a row that could not be applied to a target is written to this directory.

## Using Cascading Deletes and Triggers

Enterprise Replication supports cascading deletes and triggers that are defined on replicated tables. The following information describes how Enterprise Replication processes cascading deletes and triggers on target database servers.

## Using Cascading Deletes

If the source table and the target table define cascading deletes, the children rows that delete as a result of the cascading delete on a parent row on the source database server are also sent to the target database server. When this transaction is applied on the target database server, the cascading delete is invoked when the parent row is processed and the children rows are deleted. Subsequently, when deletes are processed on the children, an error might result because the rows were already deleted. The table in Figure 7-13 indicates how Enterprise Replication resolves cascading deletes with conflict-resolution scopes and rules.

***Figure 7-13***
*Resolving Cascade Deletes*

| Conflict-Resolution Rule | Conflict-Resolution Scope | Actions on Delete Errors | Result |
|---|---|---|---|
| Time stamp | Row-by-row or transaction | Note as replication exceptions | Process children rows |
| Ignore | Transaction | Report as errors | Abort entire transaction |
| Ignore | Row-by-row | Report as errors | Reject row |

## Using Triggers

If the trigger option is enabled, triggers that are defined on a table that is defined for replication are invoked when transactions are processed on the target database server. Similar to the cascading deletes, if the same triggers are defined on both the source and target tables, any insert, update, or delete operation that the triggers generate are also sent to the target database server. These operations can result in errors depending on the conflict-resolution rule and scope.

In addition, the following trigger operations behave differently when they execute on the target database server:

- All replicated columns are considered to be modified if the database operation is updated, including the values for some of the columns that might not have been modified. All update triggers defined on replicated columns are invoked.
- If an update trigger is defined on table **tab1** and the following command is executed:

  ```
  UPDATE tab1 SET a = a + 1
  ```

  on the source database server, the command affects all rows in table **tab1**. If any *before* action exists for the update trigger, the *before action* trigger is executed once for this statement, any *for each row* is executed once for each row processed, and any *after* action is executed once for the statement.

  However, if the same update trigger is defined on the target table and the same command is executed on the source database server, the following results occur: all *before*, *for each row*, and *after* actions are executed once for each row that the target database server is processing.

Chapter

# 8

# Enterprise Replication Menus


In This Chapter . . . . . 8-5

General Information . . . . . 8-6
- Preparing to Use the Replication Manager . . . . . 8-6
- Opening the Replication Manager . . . . . 8-6
- Understanding Server Terminology . . . . . 8-7
- Using Modes . . . . . 8-7
- Choosing Ellipses . . . . . 8-8
- Viewing Long Identifiers . . . . . 8-8

The Replication Manager Main Window . . . . . 8-10
- Using the Global Catalog Icon . . . . . 8-10
- Using the What's This Icon . . . . . 8-11

The File Menu . . . . . 8-11
- Creating Scripts . . . . . 8-11
- Monitoring Replication Events . . . . . 8-11

The Replicate Menu . . . . . 8-12
- Managing Replicates and Participants . . . . . 8-13
  - Defining a Replicate . . . . . 8-13
  - Deleting a Replicate . . . . . 8-13
  - Adding a Participant . . . . . 8-14
  - Removing a Participant . . . . . 8-14
- Changing Replicate States . . . . . 8-14
  - Active . . . . . 8-15
  - Inactive . . . . . 8-16
  - Quiescent . . . . . 8-16
  - Suspended . . . . . 8-16
- Changing the Primary Participant . . . . . 8-17

Viewing Replicate Properties . . . . . 8-17
General Properties Tab . . . . . 8-18
Conflict Resolution Tab . . . . . 8-18
Spooling Tab . . . . . 8-19
Frequency Tab . . . . . 8-19
Participant Properties . . . . . 8-19

Group Menu . . . . . 8-20
Defining a Replicate Group . . . . . 8-20
Modifying Groups . . . . . 8-20
Deleting a Replicate Group . . . . . 8-21
Adding Replicate(s) to an Existing Replicate Group . . . . . 8-21
Removing Replicate(s) from an Existing Replicate Group . . . 8-21
Changing the State of a Group . . . . . 8-22
Active . . . . . 8-22
Inactive . . . . . 8-22
Quiescent . . . . . 8-23
Suspend . . . . . 8-23
Viewing Replicate Group Properties . . . . . 8-24

Server Menu . . . . . 8-25
Suspending and Resuming a Database Server . . . . . 8-25
Finding or Changing the Server State . . . . . 8-26
Viewing the Replication Hierarchy . . . . . 8-27
Changing to Primary . . . . . 8-28
Changing to Target . . . . . 8-28
Starting and Stopping Enterprise Replication . . . . . 8-28
Stopping Replication . . . . . 8-29
Starting Replication . . . . . 8-29
Specifying Privileges . . . . . 8-29
Removing a Database Server from Enterprise Replication . . . . 8-30
Reloading the Global Catalog . . . . . 8-31
Viewing Database Server Properties . . . . . 8-31

View Menu . . . . . 8-32
By Replicate and By Server . . . . . 8-32
Toolbar and Status Bar . . . . . 8-32

Window Menu . . . . . 8-33
Nonexpert Mode . . . . . 8-33
Expert Mode . . . . . 8-34
List of Open Windows . . . . . 8-34

The Help Menu. . . . . 8-35
Help Topics. . . . . 8-35
Accessing Reference Help . . . . . 8-35
Accessing Context-Sensitive Help. . . . . 8-37
Using Help . . . . . 8-37
Contacting Informix. . . . . 8-37
About Replication Manager . . . . . 8-37

Hierarchical View Menus . . . . . 8-38

Replication Scripting View Menus . . . . . 8-39

WIN NT

# In This Chapter

The Enterprise Replication Manager (ERM) is a Windows NT graphical user interface (GUI) provided to help you administer and monitor Enterprise Replication. This chapter covers the function of each of the menu options. The on-line help provides additional information about various options. The next chapter, Chapter 9, "Enterprise Replication Manager Wizards," describes the ERM wizards.

To use ERM with Dynamic Server, Version 7.31, install Informix Enterprise Command Center (IECC). From IECC, choose **Tools→Enterprise Replication Manager**.

To use ERM with Dynamic Server, Version 9.2, install ERM and the proper connectivity tools. For information about preparing ERM for Version 9.2, refer to the documentation notes described in "Documentation Notes, Release Notes, Machine Notes" on page 15 of the Introduction. ♦

You can perform all the operations that are described in this chapter with the command-line utility (CLU) described in Chapter 11 or with the application-programming-interface (API) calls described in Chapter 12.

# General Information

This section discusses some general information that you need before you work with ERM.

## Preparing to Use the Replication Manager

Before you can use ERM, you must prepare SQLHOSTS connectivity information for the database servers and database server groups.

UNIX

The **sqlhosts** file for each database server contains connectivity information about every database server and database server group that participates in replication. For information about preparing the sqlhosts file, refer to your *Administrator's Guide*. ♦

WIN NT

With Dynamic Server, Version 7.31, use the Add Database Server wizard in Informix Enterprise Command Center (IECC) to prepare the SQLHOSTS connectivity information.

With Dynamic Server, Version 9.2, refer to the documentation notes described in "Documentation Notes, Release Notes, Machine Notes" on page 15 of the Introduction for information about preparing the SQLHOSTS connectivity information. ♦

You can also use **regedt32** to prepare SQLHOSTS information for your database servers. For more information, refer to Appendix E, "SQLHOSTS Registry Key."

## Opening the Replication Manager

When you open ERM, it checks to see whether any database servers have been defined for Enterprise Replication. If at least one replication server has already been defined, the Replication Manager main window appears, as shown in Figure 8-3 on page 8-10.

If no replication servers have been defined (that is, if you are starting replication for the first time), you must define a database server for replication before you can continue. "Defining the First Replication Server" on page 3-18 illustrates the technique for defining a replication server.

When you declare the first Enterprise Replication server, make the following choices:

- Synchronization Server

  The first server that you declare does not have a synchronization server. Make sure that the **Use a Synchronization Server** check box is *not* checked. The Synchronization Server pane should be gray.

- Queue dbspaces

  If you do not want the send and receive queues to go into the root dbspace, you must exit from ERM (choose **Cancel**), create the dbspaces, and then return to ERM.

- Spooling

  Make sure that the spooling directory exists. If the directory does not exist, the declare-server process fails.

## Understanding Server Terminology

Many of the pages in ERM ask you to select a *server*. In ERM, *server* is usually synonymous with *database server group*. Each database server that participates in a replication system must be a member of a database server group. For more information, refer to "Database Server Groups" on page 11-12.

## Using Modes

ERM has two modes, expert mode and nonexpert mode. Nonexpert mode is simpler to use because it has limited options. This chapter documents both expert and nonexpert modes. For information about the modes, refer to "Expert Mode" on page 8-34 and "Nonexpert Mode" on page 8-33.

## Choosing Ellipses

A menu item that contains an *ellipsis* (…), (for example **Define Replicate…**), indicates that a wizard, dialog box, or a property sheet follows to help you complete a task. The ERM wizards are described in Chapter 9, "Enterprise Replication Manager Wizards."

## Viewing Long Identifiers

*Identifiers* are the names of objects that Dynamic Server and Enterprise Replication use, such as database servers, databases, columns, replicates, replicate groups, and so on.

An identifier is a character string that must start with a letter or an underscore. The remaining characters can be letters, numbers, or underscores. Most identifiers can be 18 bytes long. However, replicates and replicate groups can be 128 bytes long.

For more information about identifiers, see the *Informix Guide to SQL: Syntax*.

In Informix Dynamic Server 2000 all identifiers can be 128 bytes long. However, if you have any database servers in your replication environment that are from an earlier version, you must follow the length restrictions for that version.

The location of files, such as the location of the ATS files, can be 256 bytes.

User login IDs can be a maximum of 32 bytes. The owner of a table is derived from a user ID and is thus limited to 32 bytes.

Although long identifiers for objects are permitted, most objects have reasonably short names. Therefore, the displays in the Replication Manager are designed to accommodate identifiers that are not very long. When you encounter a long name in the Replication Manager, you can click on the name to see the full name expanded or use the scroll bar to view the entire name.

For example, in Figure 8-1 the long replicate names are truncated. In Figure 8-2, the cursor is pointing to the first replicate, causing the replicate name to expand to its full length.

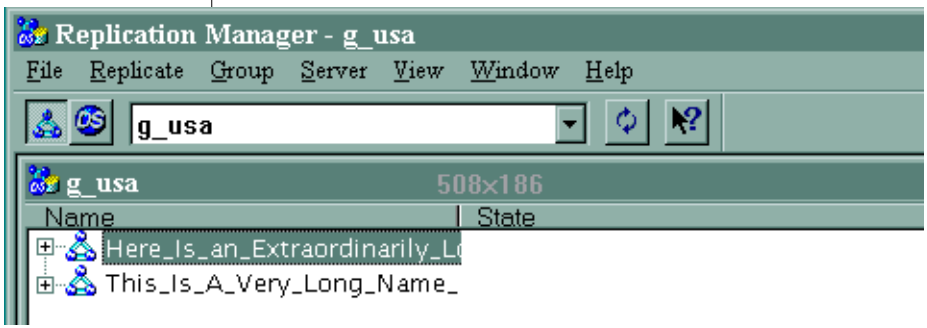
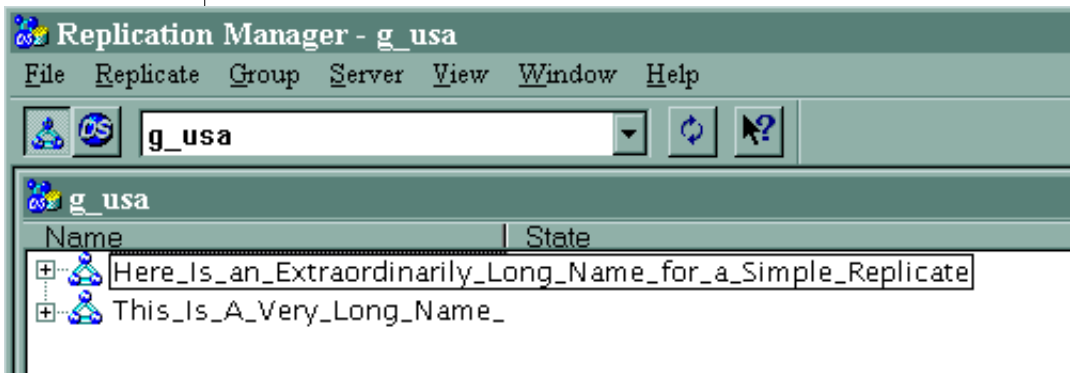



***Figure 8-1***
*Long Replicate Name, Truncated*

***Figure 8-2***
*Long Replicate Name, Expanded*



# The Replication Manager Main Window

The main window provides a central point of control for the Replication Manager. You can define, modify, delete, and view database servers, replicates and groups, and access hierarchical views, scripting views, and the event monitor. Figure 8-3 shows the header and toolbar of the main window.

To access menu items, click a menu from the title bar and then choose the menu item. You can also select a replicate, group, or database server and press the right key of your mouse to access the menu items.

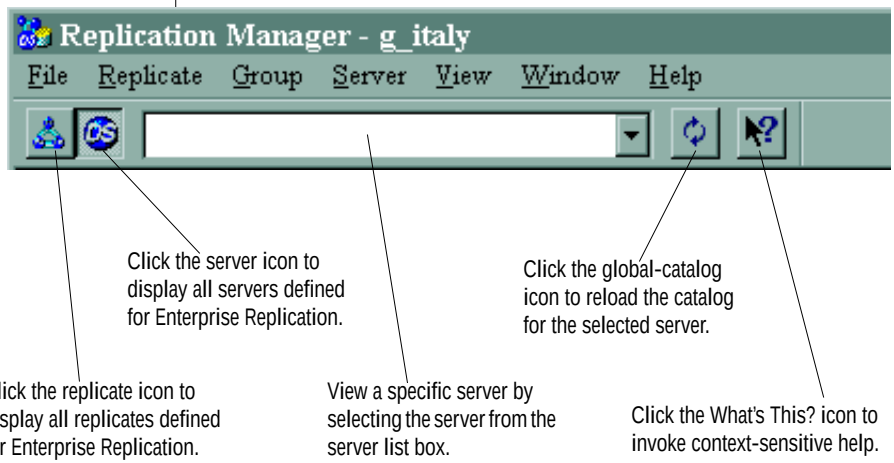

Replication Manager - g_italy
File Replicate Group Server View Window Help
Click the server icon to display all servers defined for Enterprise Replication.
Click the global-catalog icon to reload the catalog for the selected server.
Click the replicate icon to display all replicates defined for Enterprise Replication.
View a specific server by selecting the server from the server list box.
Click the What's This? icon to invoke context-sensitive help.

***Figure 8-3***
*Replication Manager Main Window*

## Using the Global Catalog Icon

When you click the global-catalog icon, ERM reloads the global catalog of the server appearing in the server list box, the currently selected server. If you have been using the CLU as well as ERM, the current copy of the global catalog might be out-of-date.

## Using the What's This Icon

When you click the **What's This?** icon, the cursor changes to an arrow with a question mark. Drag this icon to the object for which you want help and click the left mouse button.

# The File Menu

The **File** menu includes the following options:

- New Script (expert mode only)
- Open Script (expert mode only)
- Event Monitor
- Exit

## Creating Scripts

The **New Script** and **Open Script** menu options open the Replication Scripting View. This view lets you create, modify, and save different replication environments. For more information, refer to "Replication Scripting View Menus" on page 8-39.

## Monitoring Replication Events

The Event Monitor option lets you view a summary of replication errors and warnings. Enterprise Replication records warnings or severe errors that occur on a *remote* replication server; it does not record error messages from the local replication server to the Replication Event Monitor.

From the Event Monitor display, you can select a single error and ask the event monitor to show more detail about that error.You can also mark individual events as *reviewed*, delete selected events, or export the information to a text file for printing.

# The Replicate Menu

The **Replicate** menu lets you define and control replicates. In the main window, you can click the replicate icon ( ) to see replicates and participants, as Figure 8-4 illustrates.

***Figure 8-4***
*Replicate View*

To use the Replicate menu, select a replicate and then choose an item from the menu. The **Replicate** menu has the following options:

- Define Replicate...
- Delete Replicate
- Add Participant(s)...
- Remove Participant(s)
- Start Replicate (Participant)
- Stop Replicate (Participant)
- Suspend Replicate
- Resume Replicate
- Change to Primary...
- Change to Target... (expert mode only)
- Replicate Properties...
- Participant Properties...

## Managing Replicates and Participants

The first four items on the Replicate menu allow you to define and delete replicates and to add and remove participants.

### Defining a Replicate

To define a new replicate, choose **Replicate→Define Replicate**. The Define Replicate wizard appears and guides you through the steps for defining a replicate. "Defining a Replicate" on page 9-4 describes the Define Replicate wizard.

To define a replicate with the CLU, refer to "cdr define replicate" on page 11-27.

### Deleting a Replicate

To delete a replicate, select the replicate and then choose **Replicate→Delete Replicate**. ERM asks you to verify that the participant should be removed. When ERM receives this command, it immediately purges all data from the queues associated with the replicate and removes the replicate from the global catalog.

To remove a replicate with the CLU, refer to "cdr delete replicate" on page 11-39.

***Important:*** *If you delete an active replicate, Enterprise Replication cannot guarantee consistent tables. Before deleting a replicate, stop the replication and wait for the queues to empty.*

### Adding a Participant

To add participant(s) to an existing replicate, select the replicate and choose **Replicate→Add Participant(s)…**.The Add Participant wizard appears and guides you through the steps for defining a replicate. "Adding a Participant" on page 9-15 describes the Add Participant wizard.

To add a participant to a replicate with the CLU, refer to "cdr change replicate" on page 11-22.

***Warning:*** *When you add a participant to an existing replicate, you must synchronize the databases before you start replication with the new participant. For information on how to synchronize your databases, see Chapter 6, "Preparing Data for Replication."*

### Removing a Participant

To remove a participant from a replicate, click the replicate to display the participants. Select a participant and choose **Replicate→Remove Participant**. ERM asks you to verify that the participant should be removed.

To remove a participant with the CLU, refer to "cdr change replicate" on page 11-22.

## Changing Replicate States

The second group of choices on the **Replicate** menu allows you to change the state of a replicate or participant. A replicate can be in one of the following states:

- Active
- Inactive
- Quiescent
- Suspended

A participant can be either active or inactive. You cannot suspend an individual participant, only a replicate. If any participant of the replicate is active, the replicate itself is active. If all participants are inactive, the replicate is inactive.

### Active

When a replicate is in active state, Enterprise Replication captures data from the logical log and transmits the captured data to participants. To change the state of a replicate to active, select the replicate and then choose **Replicate→Start Replicate**. The Start Replicate dialog box lets you select the participants to start in the replicate, as shown in the left-hand display in Figure 8-5.

***Figure 8-5***
*Start Replicate and Stop Replicate Dialog Boxes*

If you select a participant, the **Replicate** menu changes to include **Start Participant** instead of **Start Replicate**. When you choose **Start Participant**, only the single participant that you selected becomes active.

To start a replicate or participant with the CLU, refer to "cdr start replicate" on page 11-67.

### Inactive

When a replicate is in inactive state, no database changes are captured, transmitted, or processed. When you first define a replicate, it is inactive.

To change the state of a replicate to inactive, select the replicate and then choose **Replicate→Stop Replicate**. The right-hand display in Figure 8-5 shows a Stop Replicate dialog box.

If you select a participant, the **Replicate** menu changes to include **Stop Participant** instead of **Stop Replicate**. When you choose **Stop Participant**, only the single participant that you selected becomes inactive.

To stop a replicate or a participant with the CLU, refer to "cdr stop replicate" on page 11-71.

### Quiescent

If you choose **Replicate→Stop Replicate** (or **Stop Participant**), the replicate (or participant) enters a quiescent state. No more database changes are captured for the replicate (or participant). However, activity continues until all changes on transactions that are already committed are delivered. Only complete transactions are delivered. None of the changes from a transaction that is committed after **Stop Replicate** (or **Stop Participant**) are delivered.

The quiescent state is a temporary state. You cannot control the quiescent state from the Replication Manager.

### Suspended

In suspended state, the replicate still captures and accumulates database changes but does not transmit any of the captured data. The data is queued and waits until the state is changed to active to transfer the captured data. You cannot suspend a replicate unless at least one of its participants is active.

To change the state to suspended, choose **Replicate→Suspend Replicate**.

To change a replicate from suspended state back to its previous state, choose **Replicate→Resume Replicate**. The database server transfers all data accumulated during the suspended state. After all the data is transferred, the replicate returns to active state.

***Warning:*** *When a group is suspended, Enterprise Replication holds the replication data in the send queue until the group is resumed. If lots of data is generated for the group while it is suspended, the send queue space can fill, causing data to be lost.*

***Warning:*** *Enterprise Replication does not synchronize transactions if a replicate is suspended. For example, a transaction that updates tables X and Y will be split if replication for table X is suspended.*

To suspend a replicate with the CLU, refer to "cdr suspend replicate" on page 11-75. To resume a replicate with the CLU, refer to "cdr resume replicate" on page 11-62.

## Changing the Primary Participant

***Important:*** *This option is available only in expert mode.*

One of the participants in a replicate is the primary database server. Changes on the primary database server are replicated to the other participants in the replicate. To specify a different participant as the primary database server in a replicate, select the participant that should become the primary database server and choose **Replicate→Change to Primary** from the Replicate menu.

## Viewing Replicate Properties

The Replicate Properties window summarizes the properties you have assigned to a replicate.To view replicate properties, select a replicate and then choose **Replicate→Replicate Properties**.

The Replicate Properties window has the following tabs:

- General
- Conflict Resolution (expert mode only)
- Spooling (expert mode only)
- Frequency

### General Properties Tab

In nonexpert mode, the **General** tab displays only the name of the replicate. In expert mode, the **General** tab displays the replicate name and indicates whether triggers and canonical message format are enabled, as Figure 8-6 shows.

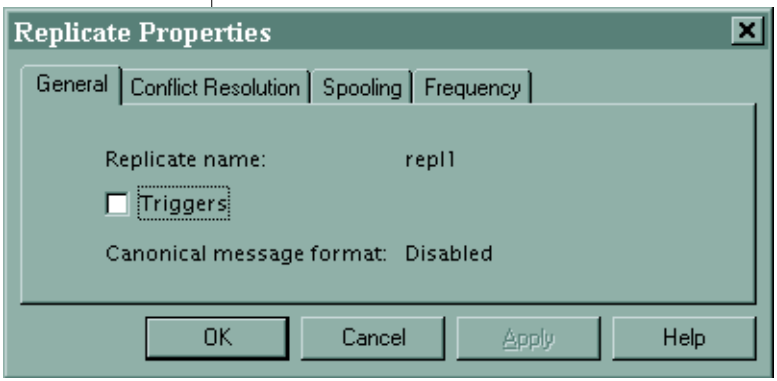



***Figure 8-6***
*Group Properties Dialog Box in Nonexpert Mode and in Expert Mode*

For information about enabling triggers, refer to Chapter 7, "Enterprise Replication Processes." For information about canonical message format, refer to "Replicate Options" on page 2-9.

### Conflict Resolution Tab

***Important:*** *This option is available only in expert mode.*

The **Conflict Resolution** tab displays the following information about the conflict-resolution information chosen for the replicate:

- Conflict resolution scope
- Conflict resolution rule
- Stored procedure

For more information, refer to "Choosing Conflict-Resolution Rules and Scopes" on page 7-13.

To find this information using the CLU, refer to "cdr list replicate" on page 11-48.

### Spooling Tab

The **Spooling** tab shows whether ATS and / or RIS spooling are active. For information about spooling, refer to Chapter 10, "Diagnosing Enterprise Replication."

To find this information using the CLU, refer to "cdr list replicate" on page 11-48.

### Frequency Tab

Click the **Frequency** tab in the **Replicate Properties** window to view or change the frequency, and then click **Apply**. For more information, refer to "Frequency Options" on page 11-14.

***Important:*** *Enterprise Replication calculates a new replication schedule at the time you select Apply. For example, if you change the time-based frequency value from every hour to every two hours and select Apply at 12:20 P.M., replication occurs at 2:20 P.M.*

### Participant Properties

The Participant Properties window shows the columns and the WHERE clause selected for the participant. To view participant properties, select a participant and then choose **Replicate→Participant Properties**. ERM displays the properties of the participant, as Figure 8-7 shows.

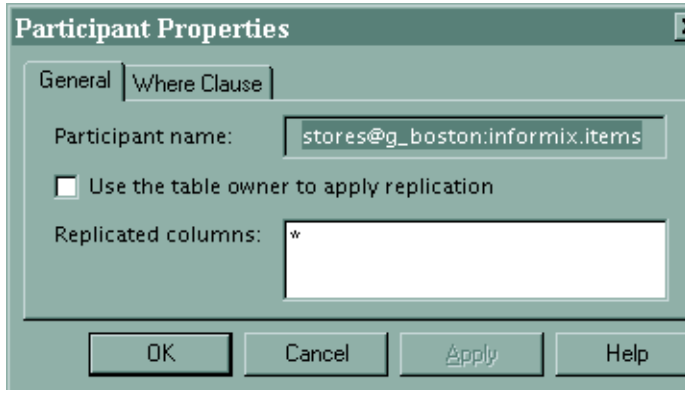



***Figure 8-7*** *Participant Properties Window*

# Group Menu

The **Group** menu lets you define and control replicate groups and review replicate group properties. All items on the **Group** menu are available in both expert and nonexpert modes. The Group menu has the following options:

- Define Group...
- Delete Group
- Add Replicate(s)...
- Remove Replicate
- Start Group
- Stop Group
- Suspend Group
- Resume Group
- Group Properties

## Defining a Replicate Group

To define a replicate group, choose **Group→Define Group**. ERM starts the Define Replicate Group wizard. The Define Replicate Group wizard provides a series of pages that guide you through the process to define a replicate group. For information about this wizard, refer to "Defining a Replicate Group" on page 9-15.

To define a replicate group using the CLU, refer to "cdr define group" on page 11-25.

**Modifying Groups.** After you define at least one replication group, you can use the choices on the **Group** menu to modify the groups.

### Deleting a Replicate Group

To delete a replicate group, select the replicate group. Choose **Group→Delete Group**. When Enterprise Replication receives this command, it deletes the replicate group from the global catalog. The **Delete Group** command does not affect the replicates or the associated data in the replicate group. If you have time-based replication enabled for the replicate group, the replicates defined to the replicate group retain the time-based frequency that was set for the replicate group.

To delete a replicate group using the CLU, refer to "cdr delete group" on page 11-38.

### Adding Replicate(s) to an Existing Replicate Group

To add replicates to an existing replicate group, select the replicate group and then choose **Group→Add Replicates**. The **Replicates** list in the Add Replicates dialog box shows all replicates that are not already included in the replicate group. Select the replicate(s) to add to the replicate group and click **Add**.

To add replicates to a group using the CLU, refer to "cdr change group" on page 11-20.

### Removing Replicate(s) from an Existing Replicate Group

To remove a replicate from an existing replicate group, select the replicate and choose **Group→Remove Replicate**.

To remove replicates from a group using the CLU, refer to "cdr change group" on page 11-20.

## Changing the State of a Group

The second group of choices on the **Group** menu allow you to change the state of a group. A group can be in one of the following states:

- Active
- Inactive
- Quiescent
- Suspended

### *Active*

A replicate group assumes an *active* state if all replicates in the replicate group are active. To change the state of a replicate group from inactive to active, choose **Group→Start Group**. When you choose **Start Group**, Enterprise Replication begins capturing data from the logical log and transmitting captured data to replicates.

To start a group using the CLU, refer to "cdr start group" on page 11-66.

### *Inactive*

When a replicate is in inactive state, no database changes are captured, transmitted, or processed. When you first define a replicate group, the replicate group assumes the state of the replicates in the replicate group. Therefore, if the replicates are inactive, the replicate group is inactive.

To change the state of a replicate group from active to inactive, choose **Group→Stop Group**. When you choose **Stop Group**, Enterprise Replication stops capturing data from the logical log.

To stop a group using the CLU, refer to "cdr stop group" on page 11-70.

### Quiescent

When you choose **Group→Stop Group**, the group enters a quiescent state. No more database changes are captured for any replicate in the replicate group. However, activity continues until all changes on transactions that are already committed are delivered. Only complete transactions are delivered. No changes from a transaction that is committed after **Stop Group** are delivered.

The quiescent state is a temporary state. You cannot control the quiescent state from the Replication Manager.

### Suspend

When you choose **Suspend Group,** the replicate group enters the *suspend* state. Suspended replicate groups still capture and accumulate database changes but do not transmit any of the captured data. To change the state to suspended, choose **Group→Suspend Group.** The data is queued and waits until the state is changed to active to transfer the captured data. To change the state to active, choose **Group→Resume Group**. All data accumulated during the suspended state is transferred and the database change capture continues and transmission resumes.

***Warning:*** *When a replicate is suspended, Enterprise Replication holds the replication data in the send queue until the group is resumed. If lots of data is generated for this replicate while it is suspended, the send queue space can fill, causing data to be lost.*

***Warning:*** *Enterprise Replication does not synchronize transactions if a replicate is suspended. For example, a transaction that updates tables X and Y will be split if replication for table X is suspended.*

## Viewing Replicate Group Properties

The Replicate Group Properties dialog box summarizes the properties that you assigned to a replicate group. To view replicate group properties, select the replicate group and then choose **Group→Group Properties**. ERM displays a dialog box that shows the properties of the replicate group, as Figure 8-8 on page 8-24 illustrates.

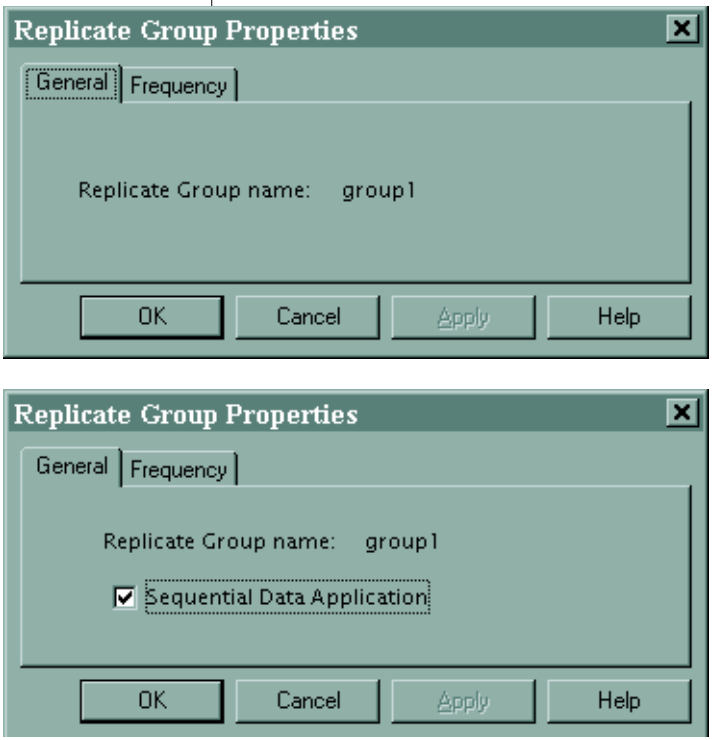



***Figure 8-8***
*Group Properties Dialog Box in Nonexpert Mode and in Expert Mode*

You can use this dialog box to change the frequency of replication. To change the frequency value, click the **Frequency** tab, change the frequency value, and then click **Apply**.

# Server Menu

Use the **Server** menu to manage the database servers that you have declared to Enterprise Replication. The **Server** menu has the following options:

- Suspend Server
- Resume Server
- Change State (expert mode only)
- Hierarchical Routing (expert mode only)
- Change to Primary
- Change to Target (expert mode only)
- Start Replication
- Stop Replication
- Use Table Owner
- Use Informix
- Remove Server
- Reload from Global Catalog
- Server Properties...

## Suspending and Resuming a Database Server

To suspend a database server, select the database server and choose **Server→Suspend Server**. **Suspend Server** suspends the delivery of replication to a database server.

To resume a database server from the **Server** menu, select the database server and choose **Server→Resume Server**. **Resume Server** resumes the delivery of replication to a database server.

To suspend or resume a server using the CLU, refer to "cdr suspend server" on page 11-77 and "cdr resume server" on page 11-63.

## Finding or Changing the Server State

***Important:*** *This option is available only in expert mode.*

This option lets you find the states of all servers in the replication environment without connecting to each individual server. Choose **Server→Change State...** to open the dialog box.

The Server State dialog box in Figure 8-9 shows that replication of data from server **usa** to server **boston** is suspended. However, replication from **boston** to **usa** is active. You can suspend replication in one direction without suspending replication in the other direction. To change the status from Suspended to Active, select the appropriate entry in the table and click **Resume**.

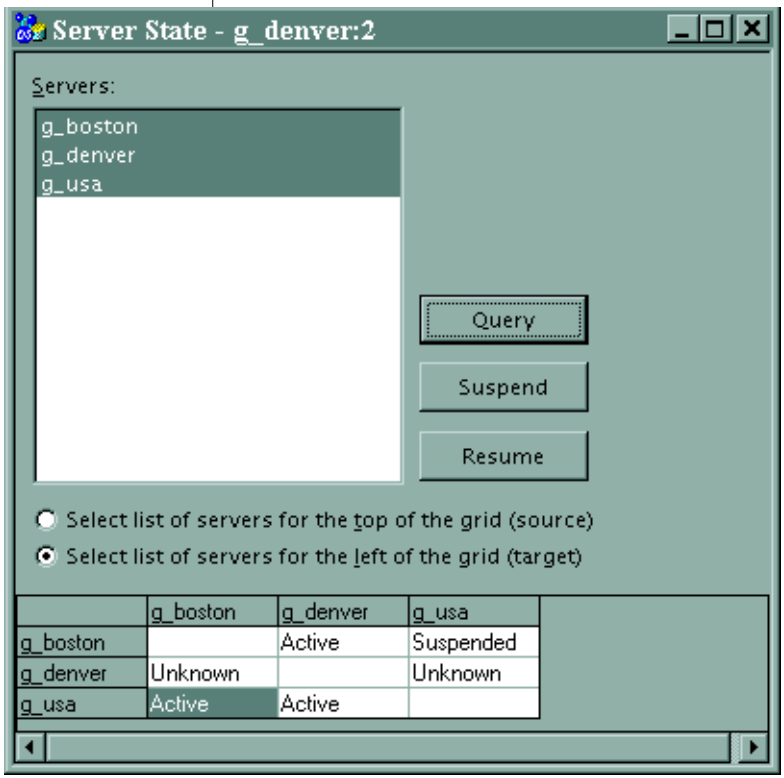



***Figure 8-9***
*Server State Dialog Box*

When you open this dialog box, only the status of replicates from **denver**, the current connection (as shown by the title bar), to the other servers is known. To find out the status of the *Unknown* entries, select the entry and then click **Query**.

To suspend or resume a server using the CLU, refer to "cdr suspend server" on page 11-77 and "cdr resume server" on page 11-63. To see the server states using the CLU, refer to "cdr list server" on page 11-50.

## Viewing the Replication Hierarchy

***Important:*** *This option is available only in expert mode.*

Select **Server→Hierarchical Routing** to open a window that shows a graphical display of the replication hierarchy. Figure 8-10 shows a hierarchical view. In this view a solid circle represents a root server, the smaller circle with a network represents a nonroot server, and the circle on two petals represents a leaf server.

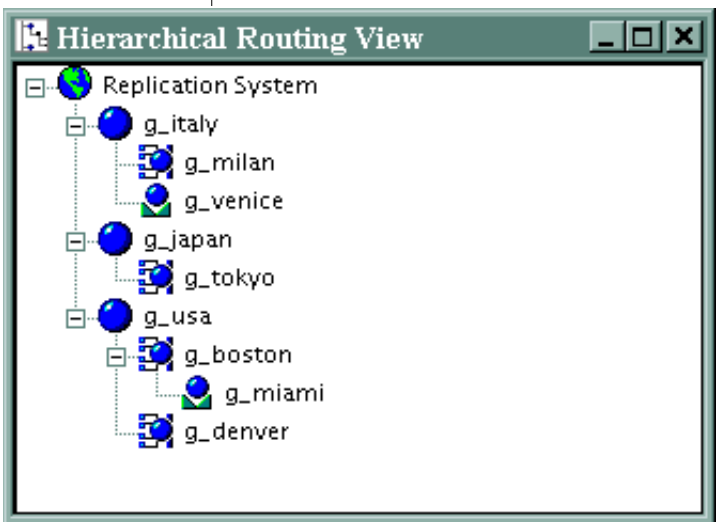



***Figure 8-10***
*Hierarchical View*

For information about the menus that are available when the Hierarchical Routing View window is active, refer to "Hierarchical View Menus" on page 8-38.

For information about Enterprise Replication topologies, refer to "Enterprise Replication Topologies" on page 2-3.

## Changing to Primary

In nonexpert mode, this option lets you change which server in a primary-target replication is the primary server. When you select a new server to be the primary server, the previous primary server becomes a target server. To change a server to primary, select a server and then choose **Server→Change to Primary**.

In expert mode, multiple primary servers can replicate information to a target server. Use this option to change a target server to a primary server.

To change a primary or secondary server with CLU, refer to "cdr modify replicate" on page 11-55.

## Changing to Target

***Important:*** *This option is available only in expert mode.*

In expert mode, a replicate can have multiple primary servers. Use **Server→Change to Target** to change one of the primary servers to a target server.

## Starting and Stopping Enterprise Replication

Starting or stopping Enterprise Replication on a database server can cause your databases to get out of sychronization. Informix recommends that you start and stop Enterprise Replication only when the databases are not in active use.

### Stopping Replication

To stop Enterprise Replication on a single database server instance, select the database server and choose **Server→Stop Replication**. Stopping Enterprise Replication shuts down all Enterprise Replication threads and frees all in-memory structures associated with that database server declared to Enterprise Replication. No global catalog tables are destroyed with this command. However, if information is not stabilized (all replication information is current and synchronized on all active replicates and replicate groups), data might become inconsistent.

Stopping a database server can be useful if you need to resynchronize all tables that are to be replicated. This command might also be appropriate when you want to stop Enterprise Replication in a database server.

To stop a replicate using CLU, refer to "cdr stop replicate" on page 11-71.

### Starting Replication

To start Enterprise Replication on a single database server, select the database server on the main window and choose **Server→Start Replication**. When you start Enterprise Replication for a database server, the global catalog tables are reread and all necessary Enterprise Replication threads restarted. If the database server from which you are starting Enterprise Replication is the primary database server, Enterprise Replication restarts the evaluation of the data marked for replication in the logical log. Database server connections are either accepted or created as required based on information in the global catalog. The start command is only effective after a **Stop Replication** command is issued.

To start a replicate using CLU, refer to "cdr start replicate" on page 11-67.

## Specifying Privileges

The **Server→Use Table Owner** and **Server→Use Informix** menu items let you specify how permission checks should be applied to replication operations. **Server→Use Table Owner** causes permission checks for *owner* to be applied to the operation (such as insert or update) that is to be replicated and to all actions fired by any triggers. **Server→Use Informix** causes all operations to be performed with the privileges of user **informix**. The default is to use the privileges of user **informix**.

To specify the permission checks using CLU, refer to "cdr define replicate" on page 11-27.

## Removing a Database Server from Enterprise Replication

***Important:*** *If you remove a database server from the replication environment before the queues are properly emptied, all data that has not been delivered for the database server might also be deleted.*

To remove a database server from Enterprise Replication, select the server and then choose **Server→Remove Server**. ERM asks you to confirm that you want to remove the database server.

If you confirm that you want to delete the replication server, ERM removes all information about the database server from all Enterprise Replication catalogs. This process can take a few minutes. After completing the deletion, ERM asks you to connect to another database server.

Removing a replication server is a two-phase project. First you must remove the links to all other replication servers and then you must remove the information about the replication server itself.

### To remove a replication server

1. Remove links to other replication servers.
   a. In the server list box, choose a server (for example, **g_usa**) that is not the one you want to remove.
   b. From the server display, select the server to remove (for example, **g_miami**).
   c. Choose **Server→Remove Server.**
2. Remove the replication server.
   a. In the server list box, choose the server that you want to remove (**g_miami**).
   b. From the server display, select the server to remove (for example, **g_miami**).
   c. Choose **Server→Remove Server.**

To remove a server using CLU, refer to "cdr delete server" on page 11-40.

## Reloading the Global Catalog

To reload the global catalog for a database server, select the database server and choose **Server→Reload from Global Catalog**. Reloading the global catalog ensures that ERM is using up-to-date information about the database server.

## Viewing Database Server Properties

To view database server properties, choose **Server→Server Properties...** to access the Server Properties dialog box.The Server Properties dialog boxes in Figure 8-11 summarizes the properties you have assigned to a database server.

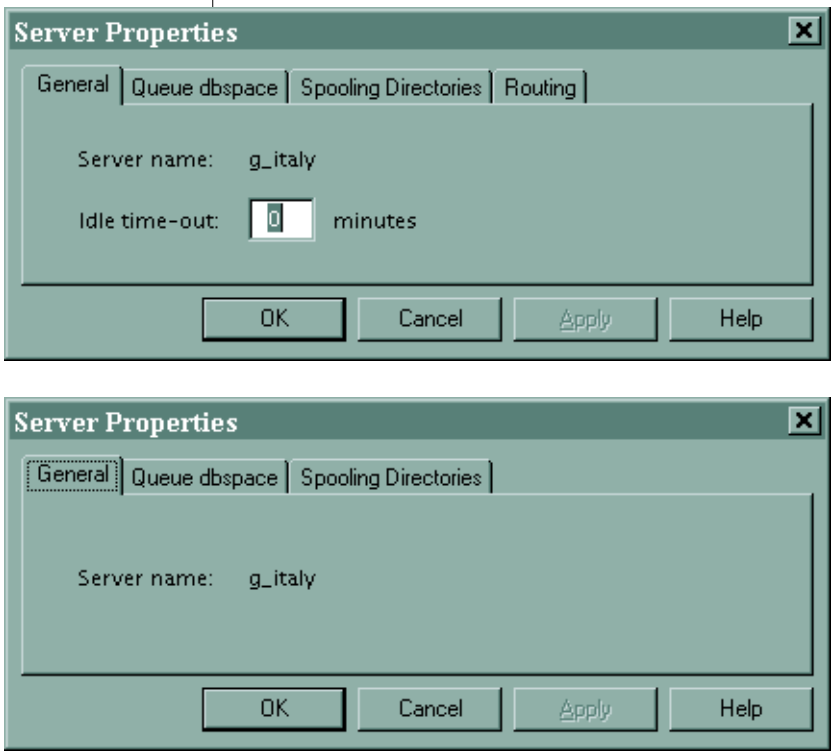



***Figure 8-11***
*Server Properties Dialog Boxes - Expert Mode and Nonexpert Mode*

Click the **General** tab to show the replication server name and, for expert mode, the idle time-out.

The **Queue dbspace** tab shows the location of the Send and Receive queues.

The **Spooling Directories** table lets you change the locations of the ATS and RIS directories.

In expert mode, the **Routing** tab gives information about the position of the replication server in the replication hierarchy.

# View Menu

The **View** menu lets you choose how information is displayed. The **View** menu has the following options:

- By Replicate
- By Server
- Toolbar
- Status Bar

## By Replicate and By Server

Use the **View** menu to view information by database server or by replicate. These menu items produce the same results as the replicate and database server icon.

## Toolbar and Status Bar

You can also choose to view (or hide) the toolbar and the status bar.

The **Toolbar** and **Status Bar** menu items cause the Toolbar and Status Bar to appear or disappear from the display.

# Window Menu

The **Window** menu lets you arrange the windows on your display, change Replication Manager modes, and activate different windows. Use the **Window** menu to arrange your windows on your workstation. The following options appear on the **Window** menu.

| Option | Function |
|---|---|
| **New List Window** | Creates a new window. This option is helpful if you want to view different information simultaneously. For example, you can show database server states in one window and replicate states in another window. |
| **Cascade** | Cascades the windows. That is, it overlaps windows on top of each other in a hierarchical order. |
| **Tile** | Tiles the windows. That is, it places the windows next to each other with no overlap. |
| **Arrange Icons** | Arranges the icons for the minimized windows at the bottom of the main window. |
| **Expert Mode** | Toggles between expert and nonexpert mode. If the choice has a check mark beside it, you are in expert mode. |
| **Database server name:*number*** | Activates the selected window. Each time you create a new window, ERM adds a numbered entry to the list of windows. To switch between windows, select an item from the list. |

## Nonexpert Mode

When ERM is first installed, it is always in nonexpert mode. Nonexpert mode has the following characteristics:

- Primary-target replication is the only replication type.
- All replication servers must be fully-connected.
- ERM assumes that all servers are on Windows NT.

In primary-to-secondary, or *primary-target* replication, one database server is the primary replication server. All changes to the primary, or source, database server are replicated to the secondary, or target, database servers. Changes to the secondary database servers are *not* replicated to the primary database server.

In a fully-connected topology, all replication servers are connected to all other replication servers. That is, all replication servers are root servers. Nonexpert mode does not allow for a hierarchical, or tree, topology.

Nonexpert mode assumes that all of your database servers are on Windows NT systems. If you have UNIX database servers, you must be careful to change the default pathnames to UNIX pathnames.

## Expert Mode

Expert mode increases the number of options that the Replication Manager menu offers and lets you choose additional types of replications. Expert mode has the following features that are not available in nonexpert mode:

- Update-anywhere replication
- Secondary-to-primary replication
- Hierarchical routing
- Scripting
- Generic configuration wizards

## List of Open Windows

Each time you open a new document window, ERM adds an entry to the numbered list at the bottom of the **Window** menu. To activate a specific window, choose its entry from the numbered list.

# The Help Menu

The **Help** menu provides access to the on-line help and information about Enterprise Replication. The **Help** menu has the following options:

- Help Topics
- Using Help
- Contacting Informix
- About Replication Manager

## Help Topics

The Replication Manager provides on-line help for both reference help and context-sensitive help. Reference help includes topics such as *What is a Replicate?* and *Defining a Primary Replicate*. Context-sensitive help provides a concise answer that describes the function of a menu item or icon that you are asking about.

### *Accessing Reference Help*

To access reference help, choose **Help→Help Topics**. Figure 8-12 shows an example of a Help Topics dialog box.

***Figure 8-12***
*Help Topics Dialog Box*

### Accessing Context-Sensitive Help

At any time, you can choose one of the following options to obtain help:

- Click the **What's This?** icon in the title bar.
- Select an object and press F1.

ERM provides additional information for all toolbar objects. When you point to an object on the toolbar, information about that object automatically appears in the status bar.

## Using Help

**Using Help** is a user's guide to **Windows Help**.

## Contacting Informix

The **Contacting Informix** option contains information on how to contact Informix Technical Support and other departments.

## About Replication Manager

The **About Replication Manager** option displays the Version Number, Serial Number, name of the principal contact and the licensing organization, and Informix trademark information. It also lists special rights and regulations with respect to government regulations.

# Hierarchical View Menus

When a Hierarchical Routing View window is active, the following menus are available.

| Menu | Options |
|---|---|
| **File** | Lets you print the hierarchy display or save a text report that lists the servers, their types, and their parent servers. The menu also lets you close the window or exit from ERM. |
| **Edit** | Lets you search for a specific replication server. |
| **Server** | Lets you connect to or disconnect from the selected server. It also shows the same Server Properties dialog box as the Server→**Server Properties** choice from the ERM main window. |
| **View** | Lets you expand the hierarchical view to show all the replication servers or collapse view to one line. The menu also lets you display or suppress the Toolbar and the Status Bar. |
| **Window** | Offers the same options as the Window menu of the ERM main window, except that it does not let you toggle between expert and nonexpert modes. |
| **Help** | Offers the same options as the Help menu of the ERM main window. |

# Replication Scripting View Menus

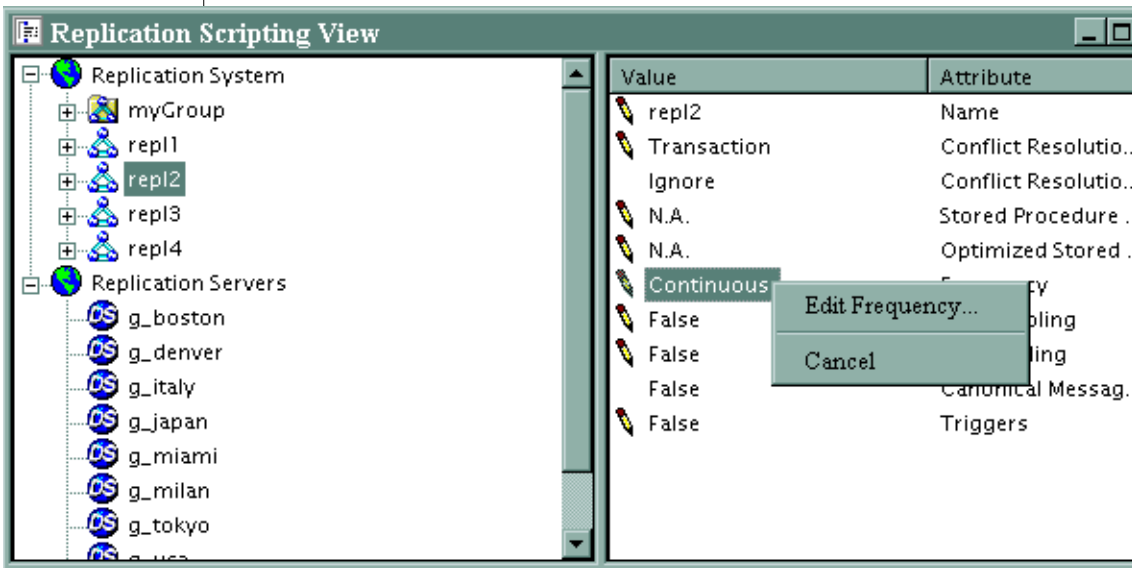
When a Replication Scripting View window is active, the following menus are available.

| Menu | Options |
|---|---|
| **File** | Lets you open, save, merge, or print replication scripts. |
| **Object** | Allows you to select an object (replicate, participant, group, or server) to add to the script, to rename an object, or to use the Generic Configuration Wizard. For information about the Generic Configuration Wizard, refer to "Generic Configuration Wizard" on page 9-17. |
| **Edit** | Lets you search for a specific replication server. |
| **View** | Lets you expand or collapse the display and switch between different display formats. The menu also lets you display or suppress the Toolbar and the Status Bar. |
| **Window** | Offers the same options as the Window menu of the ERM main window, except that it does not let you toggle between expert and nonexpert modes. |
| **Help** | Offers the same options as the Help menu of the ERM main window. |

When you open the Replication Scripting View, ERM displays a window with two panes. The left-hand pane shows the replication environment that you are scripting. That environment can be a copy of your current replication environment, an environment that you previously saved, or a new environment.

When you select an object, the right-hand pane displays the characteristics of the objects. The little pencil icon shows which items you can change. To change an item, right-click on the item and choose the action to take. Figure 8-13 shows a sample scripting view.

***Figure 8-13***
*Replication Scripting View*



Chapter

# 9

# Enterprise Replication Manager Wizards


In This Chapter . . . . . 9-3

Defining a Replicate . . . . . 9-4
- Defining a Replicate Name . . . . . 9-4
- Choosing a Conflict-Resolution Procedure . . . . . 9-5
- Choosing Replicate Options . . . . . 9-6
- Selecting the Replication Frequency . . . . . 9-7
- Selecting Servers to Participate in the Replicate . . . . . 9-9
- Specifying Identical or Non-Identical Participants . . . . . 9-10
- Defining Participant Attributes . . . . . 9-12
- Choosing the Columns to Replicate . . . . . 9-13
- Reviewing Replicate Summary . . . . . 9-14

Adding a Participant . . . . . 9-15

Defining a Replicate Group . . . . . 9-15
- Choosing a Group Name . . . . . 9-16
- Choosing the Replication Interval for the Group . . . . . 9-16
- Selecting Replicates for the Replicate Group . . . . . 9-17

Generic Configuration Wizard . . . . . 9-17
- Choosing a Naming Scheme . . . . . 9-18
  - Naming Update-Anywhere Replicates . . . . . 9-18
  - Naming Primary-Target Replicates . . . . . 9-19
- Describing the Replicates . . . . . 9-20
- Describing the Participants . . . . . 9-21
- Customizing the Generic Replicates . . . . . 9-23

WIN NT

# In This Chapter

This chapter documents the ERM wizards and guides you through some basic tasks. The previous chapter, Chapter 8, "Enterprise Replication Menus," covers the ERM menu options. The on-line help provides additional information about various options.

You can perform all the operations that are described in this chapter with the command-line utility (CLU) described in Chapter 11 or with application programming interface (API) calls described in Chapter 12.

All wizards include command buttons at the bottom of a window to allow you to navigate through the wizard. The Replication Manager uses the following command buttons.

| Command | Action |
|---|---|
| **<Back** | Returns to the previous page |
| **Next>** | Moves to the next page in sequence, keeping changes from previous pages |
| **Cancel** | Discards any changes and terminates the process |
| **Create** | Create an object that matches the description |
| **Finish** | Completes the task |
| **Help** | Requests help |

# Defining a Replicate

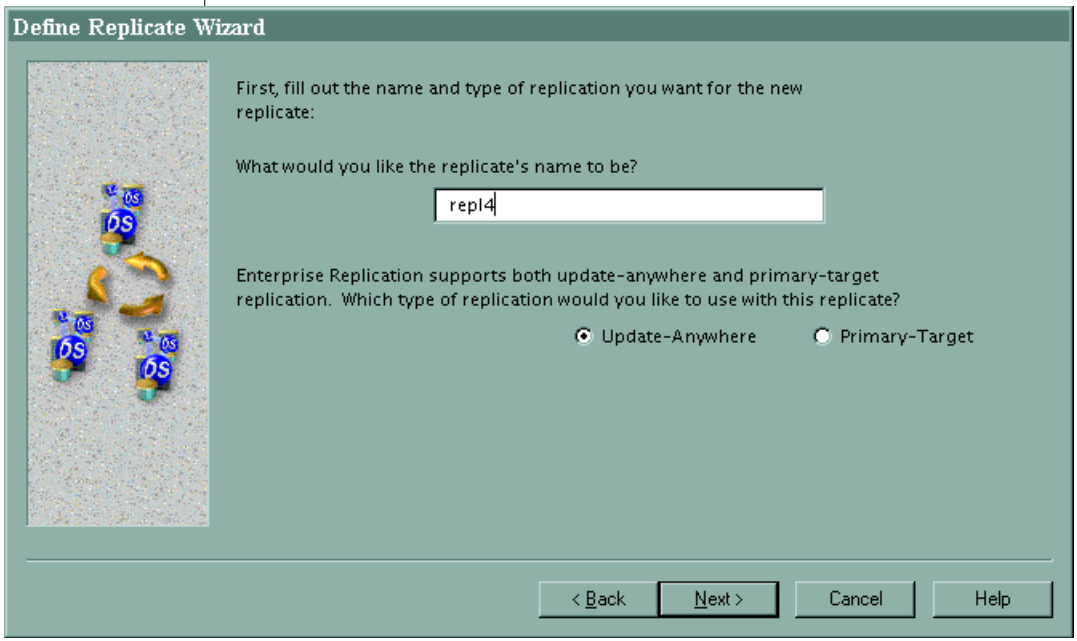
To define a replicate, choose **Replicate→Define Replicate...** The **Define Replicate...** menu item accesses the Define Replicate wizard. The wizard provides a series of windows that guide you through the process of defining a replicate.

The Define Replicate wizard first lists the steps required to define a replicate and then asks you to choose a name for the replicate.

## Defining a Replicate Name

If you are in expert mode, the wizard asks for a replicate name and replicate type, as Figure 9-1 shows. If you are in nonexpert mode, the wizard does not ask for the replicate type because only primary-target replication is available in that mode.

***Figure 9-1***
*Modifying a Generic Replicate*

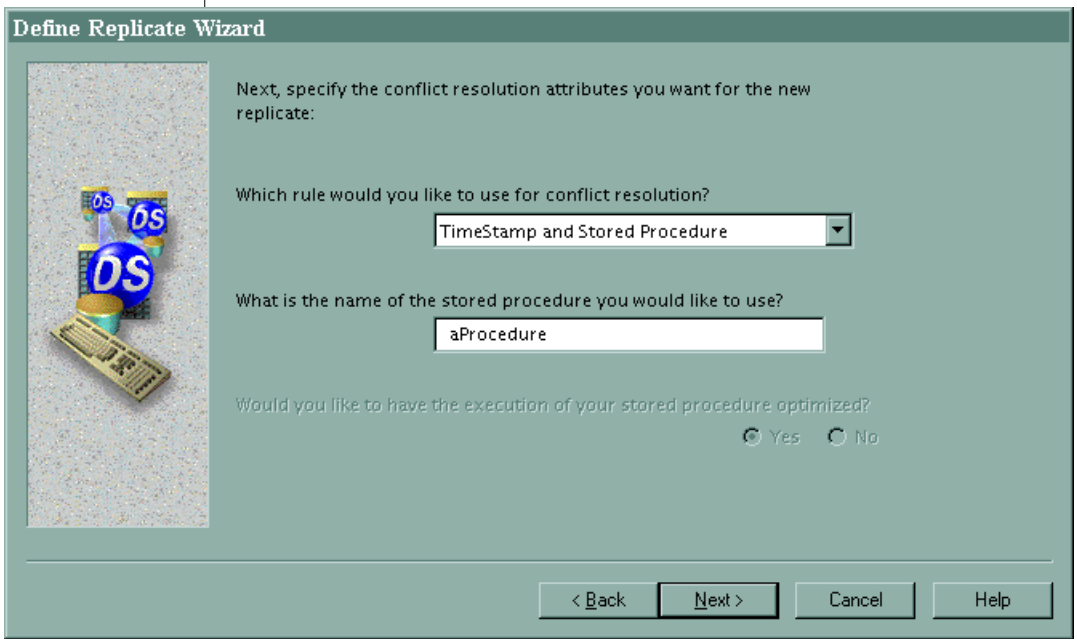

Define Replicate Wizard
First, fill out the name and type of replication you want for the new replicate:
What would you like the replicate's name to be?
repl4
Enterprise Replication supports both update-anywhere and primary-target replication. Which type of replication would you like to use with this replicate?
Update-Anywhere
Primary-Target
< Back
Next >
Cancel
Help

## Choosing a Conflict-Resolution Procedure

If you are defining a primary-target replicate, the wizard goes directly to "Choosing Replicate Options" on page 9-6. However, for update-anywhere replication you must choose a conflict-resolution procedure.

Figure 9-2 shows Timestamp and Stored-Procedure conflict resolution. When you specify a stored procedure in the wizard, you must have already written and registered the stored procedure. You cannot enter a procedure name and then create the procedure later. For information about creating stored procedures, refer to the *Informix Guide to SQL: Syntax*.

***Figure 9-2***
*Timestamp and Stored-Procedure Conflict Resolution*


Define Replicate Wizard
Next, specify the conflict resolution attributes you want for the new replicate:
Which rule would you like to use for conflict resolution?
TimeStamp and Stored Procedure
What is the name of the stored procedure you would like to use?
aProcedure
Would you like to have the execution of your stored procedure optimized?
Yes
No
< Back
Next >
Cancel
Help

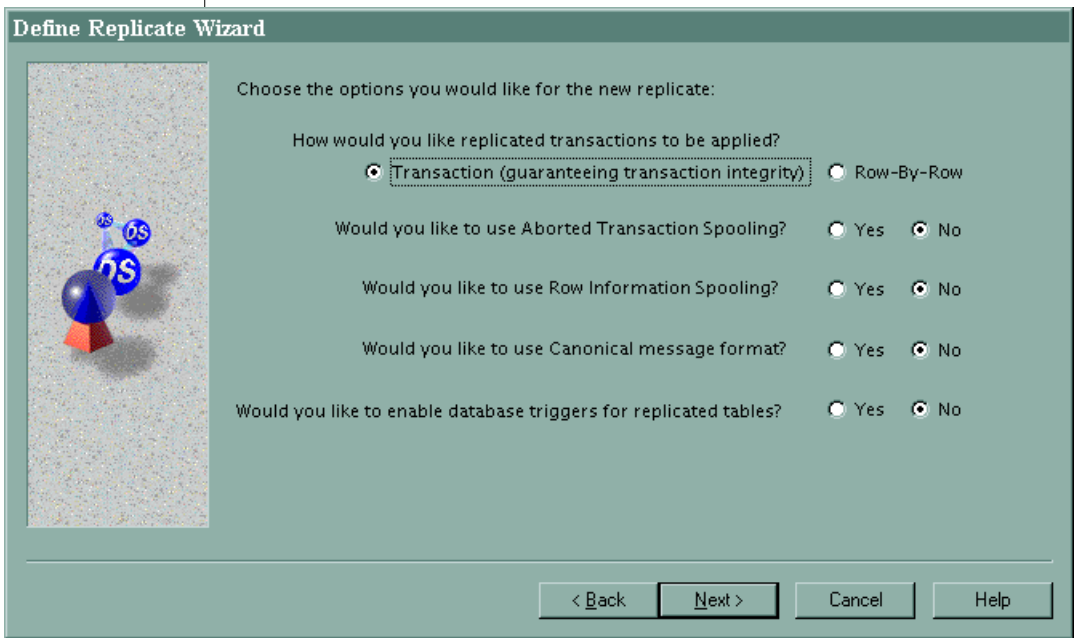
The option to optimize the stored procedure is grayed out in Figure 9-2 because you can optimize the stored procedure only when you use the procedure without the timestamp.

For more information about conflict resolution, refer to "Choosing Conflict-Resolution Rules and Scopes" on page 7-13.

## Choosing Replicate Options

The wizard next asks you to set options for the new replicate. Figure 9-3 shows the default options. For more information about the options, refer to "Viewing Replicate Properties" on page 8-17.

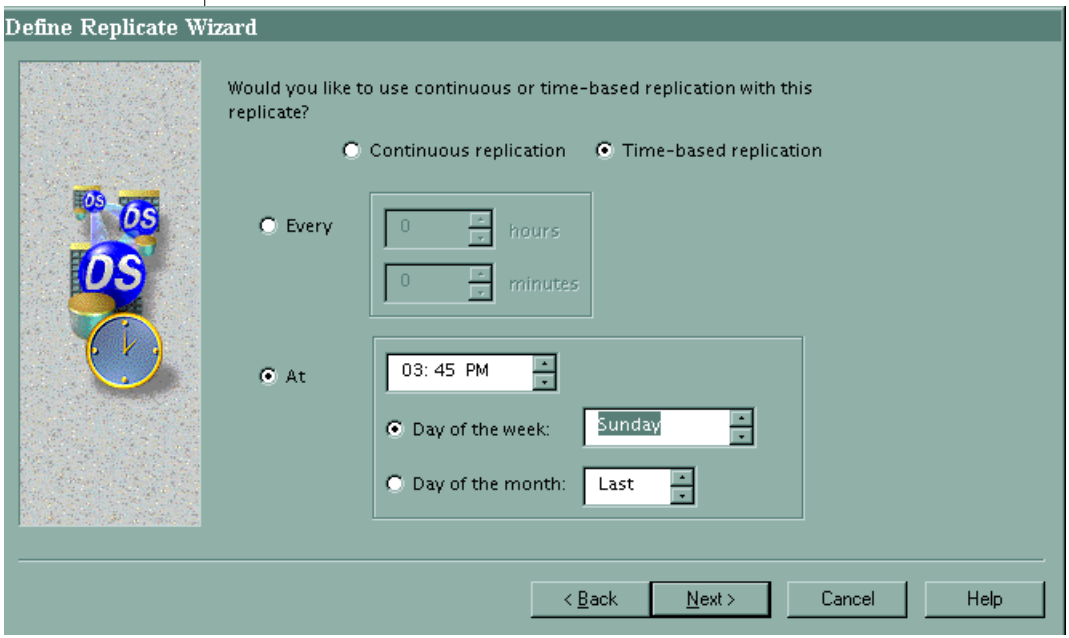
***Figure 9-3***
*Replicate Options Wizard Page*



## Selecting the Replication Frequency

Next the wizard asks you to specify whether you want replication to be continuous or time-based. In the example in Figure 9-4, time-based replication will occur every Sunday at 3:45 P.M.

***Figure 9-4***
*Select Frequency Page*



With the time-based option, you can choose the time (in hours and minutes) and day (day of week, day of month) that data replication occurs in a replicate. Time-based replication begins when you start the replicate. The following table summarizes the options you can select to determine when replication occurs.

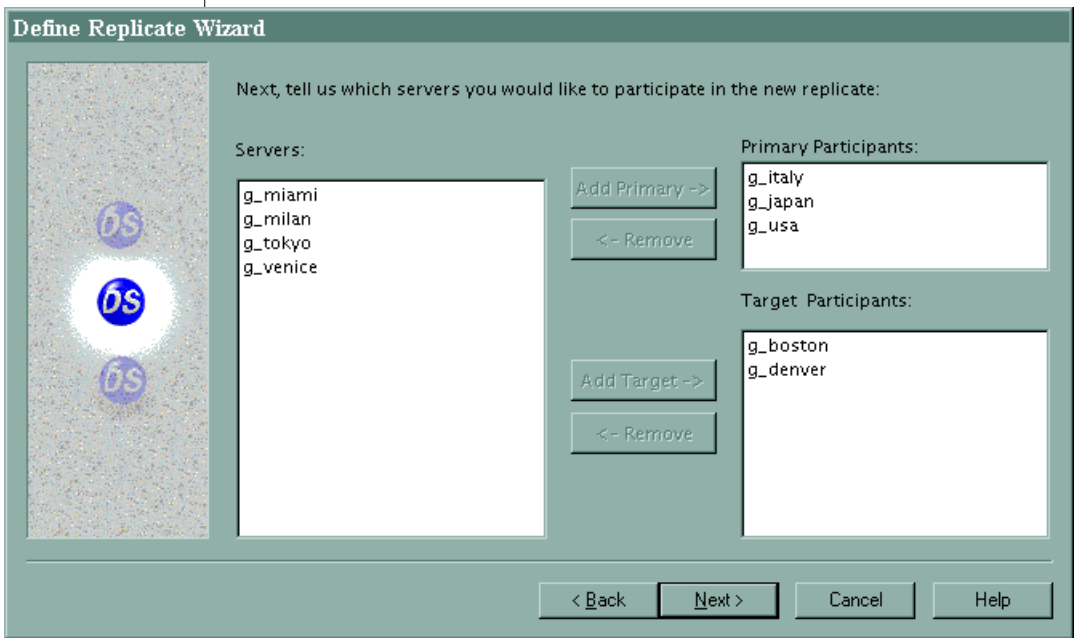
| Value | Description |
|---|---|
| Continuous | This value is the default. Enterprise Replication sends changes that should be replicated as soon as it detects them. |
| Time-based | Replicate data at a specified interval using the Every option or at a specific time on a specific day using the At option. |
| Day of the week | Replicate data once a week at the time and day specified.<br>You can also replicate data at the same time each day of the week by selecting the value `Daily` in the **Day of the week** text box. |
| Day of the month | Replicate data once a month at the specified time and day of the month. To ensure that replication occurs on the last day of every month, select the word `Last` in the **Day of month** text box.<br>If you select `31`, replication takes place *only* in months that have 31 days. |

***Important:*** *Time-based replication is based on the local time, not Greenwich mean time (GMT). If database servers participating in the replicate are in different time zones than the database server on which the time is set, replication will occur at different times.*

## Selecting Servers to Participate in the Replicate

Figure 9-5 shows a select-server wizard page. This sample page is creating a primary-target replicate in expert mode. In nonexpert mode, only one server could be chosen as the primary. The select-server page for update-anywhere is similar to Figure 9-5, except that you do not need to specify which servers are primary or secondary, because all servers in an update-anywhere replication have equal status.

***Figure 9-5***
*Select Servers Page*




The **Servers** list box lists all replication servers in the replication environment.To select the servers for your replicate, click the server name (or several server names) and click **Add**.

All database servers in an update-anywhere replicate must have read / write capabilities to allow peer-to-peer replication.

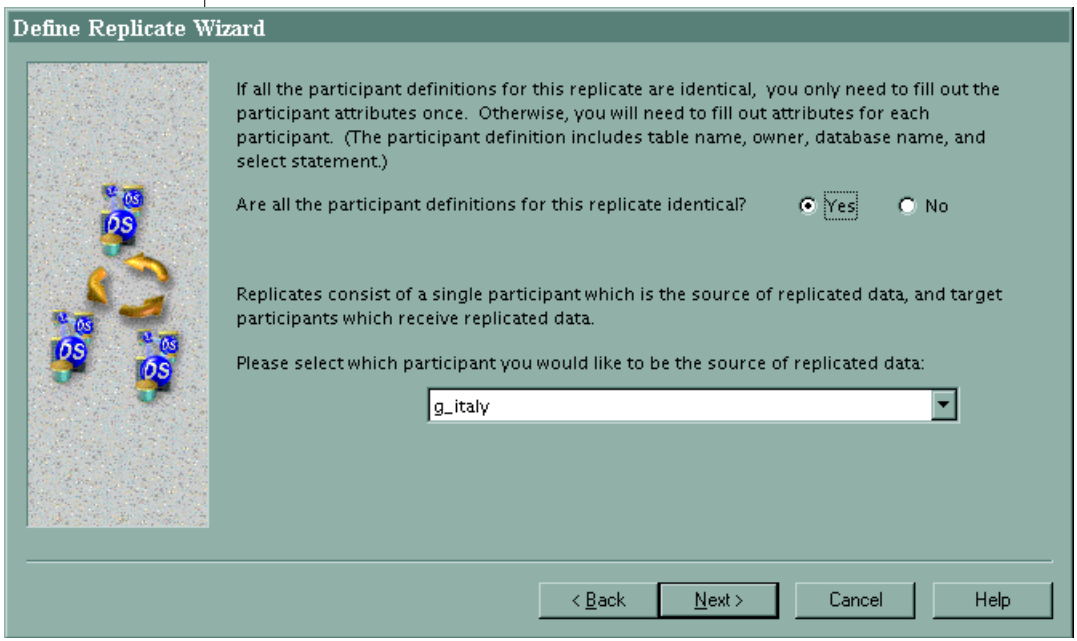
## Specifying Identical or Non-Identical Participants

After you select the database servers that will participate in the replicate, the Define Replicate wizard asks whether the attributes of all participants are identical. The participant attributes are the table name, owner of the table, database in which the tables resides, and the SELECT statement. If the attributes are identical for all participants, you will only need to specify them once. If the attributes are not identical, you must define each participant separately.

For example, if user **informix** owns (identical) **stores** databases on all database servers in the replicate, and you want to replicate the same columns on each **customers** table, then the participant attributes are identical. If, on the other hand, the databases in the replicate have identical columns, but one database is named **mansions** while the other database is named **villas**, then the attributes are not identical.

Figure 9-6 shows the identical attributes query page for nonexpert mode. In nonexpert mode, only one participant can be the primary. In expert mode, several participants can be primaries, as Figure 9-5 on page 9-9 illustrates.

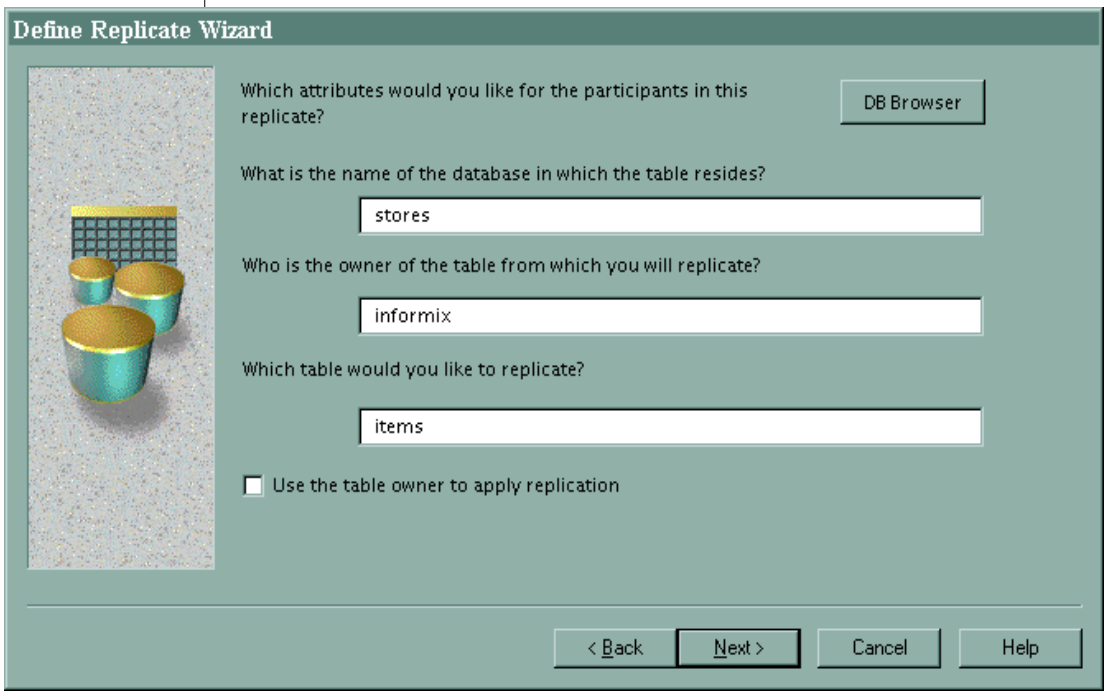
***Figure 9-6***
*Identical Attributes Page*


Define Replicate Wizard
If all the participant definitions for this replicate are identical, you only need to fill out the participant attributes once. Otherwise, you will need to fill out attributes for each participant. (The participant definition includes table name, owner, database name, and select statement.)
Are all the participant definitions for this replicate identical?
Yes
No
Replicates consist of a single participant which is the source of replicated data, and target participants which receive replicated data.
Please select which participant you would like to be the source of replicated data:
g_italy
< Back
Next >
Cancel
Help

## Defining Participant Attributes

The wizard page in Figure 9-7 asks you to define the table you want to replicate, the owner of the table, the name of the database in which the table resides, and whether to use the table owner to apply permissions.

***Figure 9-7***
*Participant Attributes Page*

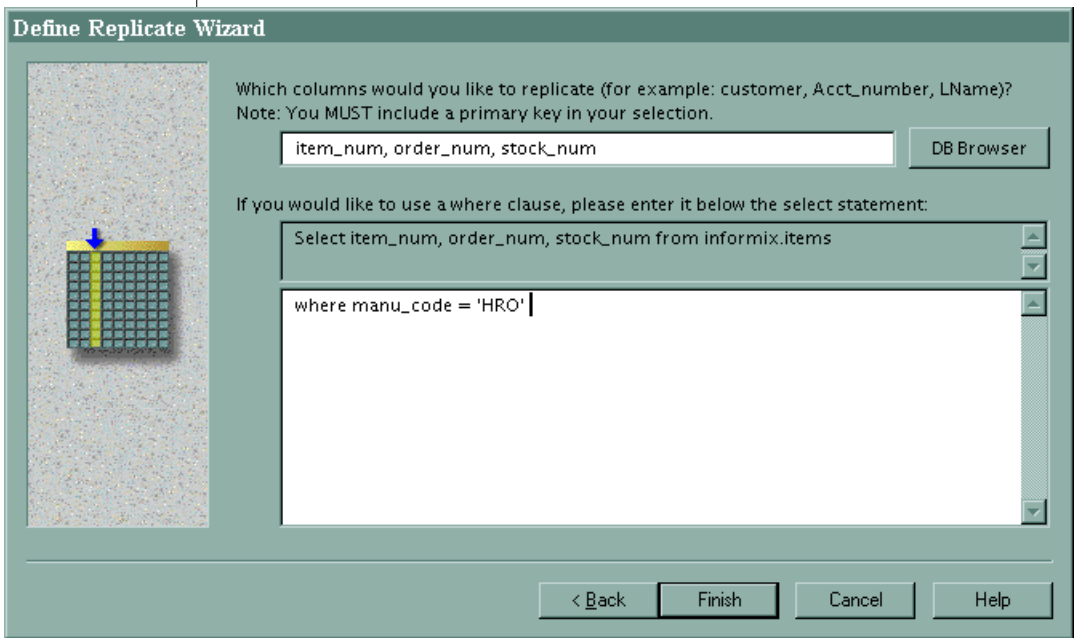

Define Replicate Wizard
Which attributes would you like for the participants in this replicate?
DB Browser
What is the name of the database in which the table resides?
stores
Who is the owner of the table from which you will replicate?
informix
Which table would you like to replicate?
items
Use the table owner to apply replication
< Back
Next >
Cancel
Help

The **DB Browser** button opens a **Database Query** window that allows you to browse through the databases of the database server of the participant. You can use the **Database Query** window to select the database, table, and columns for the participant.

## Choosing the Columns to Replicate

The wizard page in Figure 9-8 lets you select columns for replication and, if desired, restrict the data replicated with a WHERE clause. You can also use the **DB Browser** button on this page to browse through databases and select columns.

***Figure 9-8***
*Select Columns Page*


Define Replicate Wizard
Which columns would you like to replicate (for example: customer, Acct_number, LName)?
Note: You MUST include a primary key in your selection.
item_num, order_num, stock_num
DB Browser
If you would like to use a where clause, please enter it below the select statement:
Select item_num, order_num, stock_num from informix.items
where manu_code = 'HRO'
< Back
Finish
Cancel
Help

You can perform column mapping with your column selection. Each participant in the replicate must define the identical number of columns (n to n mappings) and the data types must be the same. For more information, see "Column Mapping" on page D-5.

## Reviewing Replicate Summary

After you finish entering information about the participants in the wizard, the ERM shows a summary of the attributes of the replicate that you just defined, as Figure 9-9 shows. Use the scroll bar to review all information for each database server that participates in your replicate.

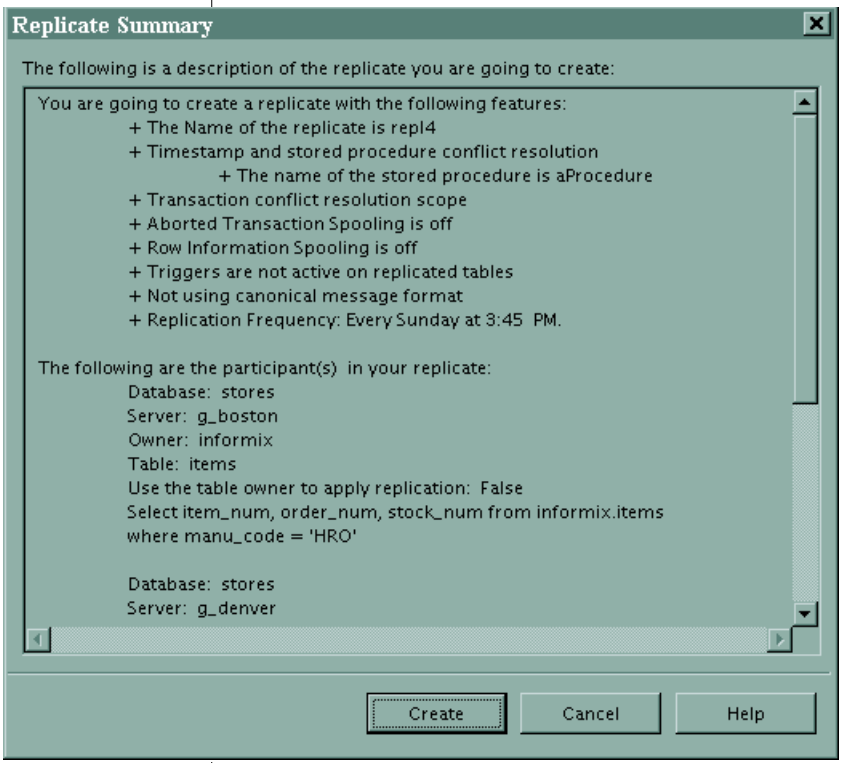

Replicate Summary
The following is a description of the replicate you are going to create:
You are going to create a replicate with the following features:
+ The Name of the replicate is repl4
+ Timestamp and stored procedure conflict resolution
+ The name of the stored procedure is aProcedure
+ Transaction conflict resolution scope
+ Aborted Transaction Spooling is off
+ Row Information Spooling is off
+ Triggers are not active on replicated tables
+ Not using canonical message format
+ Replication Frequency: Every Sunday at 3:45 PM.
The following are the participant(s) in your replicate:
Database: stores
Server: g_boston
Owner: informix
Table: items
Use the table owner to apply replication: False
Select item_num, order_num, stock_num from informix.items
where manu_code = 'HRO'
Database: stores
Server: g_denver
Create
Cancel
Help

***Figure 9-9***
*Replicate Summary Window*

If you discover errors, click **Cancel**. The wizard returns to the Define Replicate wizard, where you can make the necessary change(s). If the summary is satisfactory, click **Create** to create the replicate.

## Adding a Participant

To add participant(s) to an existing replicate, select the replicate from the main window of the Replication Manager and choose **Replicate→Add Participant(s)….**

The Add Participant wizard page lists the steps necessary to add participant(s) to an existing replicate. The Add Participant wizard continues with the steps to add a participant that parallels the steps to define a replicate in Figure 9-5 on page 9-9, Figure 9-7 on page 9-12 and Figure 9-8 on page 9-13.

***Warning:*** *If you add a participant to an existing replicate, you must synchronize the databases before you start replication with the new participant. For information on how to synchronize your databases, see Chapter 6, "Preparing Data for Replication."*

## Defining a Replicate Group

To define a replicate group, choose **Group→Define Group...**. The **Group→Define Group...** menu item starts the Define Replicate Group wizard. The wizard describes the steps needed to define a replicate group and then provides a series of pages that guide you through the process.

## Choosing a Group Name

In Figure 9-10 on page 9-16, the wizard asks you to choose a name for the replicate group and to specify whether or not to apply the replicated data sequentially. For information about sequential data application, refer to "Replicate Group" on page 2-12.

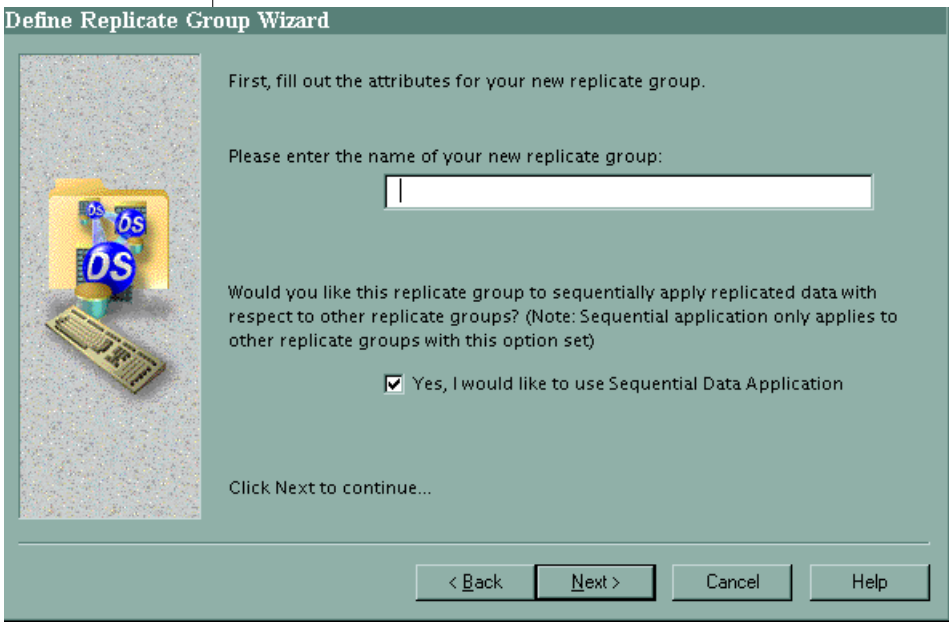

Define Replicate Group Wizard
First, fill out the attributes for your new replicate group.
Please enter the name of your new replicate group:
Would you like this replicate group to sequentially apply replicated data with respect to other replicate groups? (Note: Sequential application only applies to other replicate groups with this option set)
Yes, I would like to use Sequential Data Application
Click Next to continue...
< Back
Next >
Cancel
Help

***Figure 9-10*** *Define Replicate Group Name*

## Choosing the Replication Interval for the Group

Next the wizard asks you to choose a replication interval for your replicate group. The display for specifying the replication interval for the group is the same as the display for specifying the replication interval for a replicate, as shown in Figure 9-4 on page 9-7.

***Important:*** *Each replicate in a replicate group must have the same time-based replication frequency value (or all replicates must have continuous replication).*

## Selecting Replicates for the Replicate Group

The Define Replicate Group wizard page in Figure 9-11 asks you to select the replicates you want to participate in the replicate group. The **Replicates** box lists all defined replicates. Select the replicates for your replicate group.

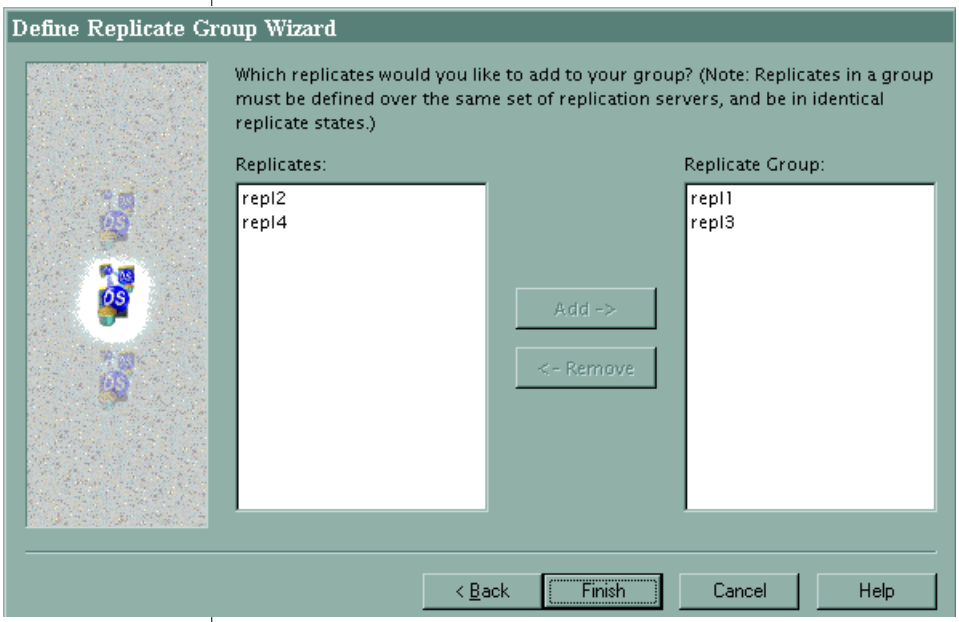



***Figure 9-11***
*Select Replicates for the Replicate Group*

When you have selected all the replicates for the group, click **Finish**. The wizard creates the replicate group for you.

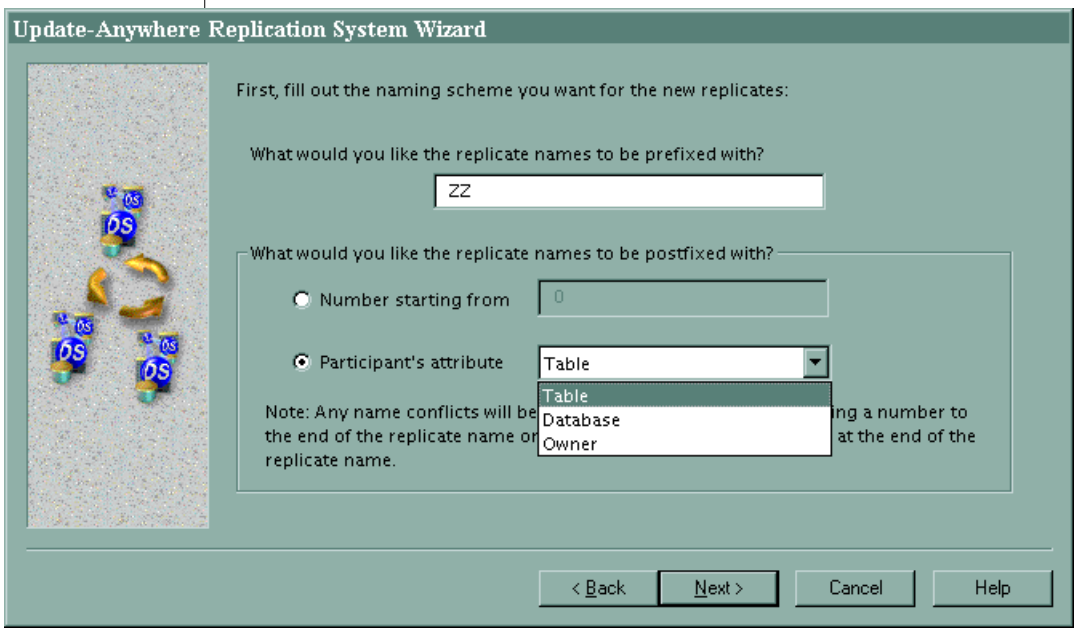
# Generic Configuration Wizard

The Generic Configuration wizard lets you create several replicates that have the same characteristics at the same time. This saves you from the work of creating each replicate individually. To access the Generic Configuration wizard, choose **File→New Script→Object→Generic Configuration Wizards.** You can create either update-anywhere replicates or primary-target replicates.

## Choosing a Naming Scheme

The wizard asks you to choose a naming scheme for the replicates that it will create. Generated replicate names can be up to 128 bytes long. The name must start with an alphanumeric character and contain only alphabetic and numeric characters and underscores. Figure 9-12 shows the naming dialog box for update-anywhere replicates.

***Figure 9-12***
*Update-Anywhere Naming Dialog Box*




### *Naming Update-Anywhere Replicates*

For generic update-anywhere replicates, the generated names have two parts, a prefix and a postfix. The prefix is an arbitrary string that identifies this group of replicates. It must start with an alphanumeric character. The postfix can be a number or an attribute of the participant.

When you select an attribute for the postfix, the wizard separates the prefix from the postfix with an underscore. When you choose a number for the attribute, the wizard does not put an underscore between the prefix and postfix. If you choose **ZZ** for the prefix, table for the attribute or 11 for the starting number, the following table shows the generated names for replicates on the **orders**, **items**, **stock**, and **state** tables.

| Using an Attribute | Using a Numeric Postfix |
|---|---|
| ZZ_orders | ZZ11 |
| ZZ_items | ZZ12 |
| ZZ_stock | ZZ13 |
| ZZ_state | ZZ14 |

## Naming Primary-Target Replicates

For generic primary-target replicates the generated names can have two or three parts: a prefix and two attributes, or a prefix and a number. In the following example, **AA** is the prefix and **g_usa** is the primary server. The left-hand column shows the generated names for replicates on the **orders**, **items**, **stock**, and **state** tables, if you choose *primary participant* for the first attribute and *table* for the second attribute. If instead you choose to number the replicates starting with 101, the right-hand column of the table shows the generated names.

| Using an Attribute | Using a Numeric Postfix |
|---|---|
| AA_g_usa_orders | AA101 |
| AA_g_usa_items | AA102 |
| AA_g_usa_stock | AA103 |
| AA_g_usa_state | AA104 |

## Describing the Replicates

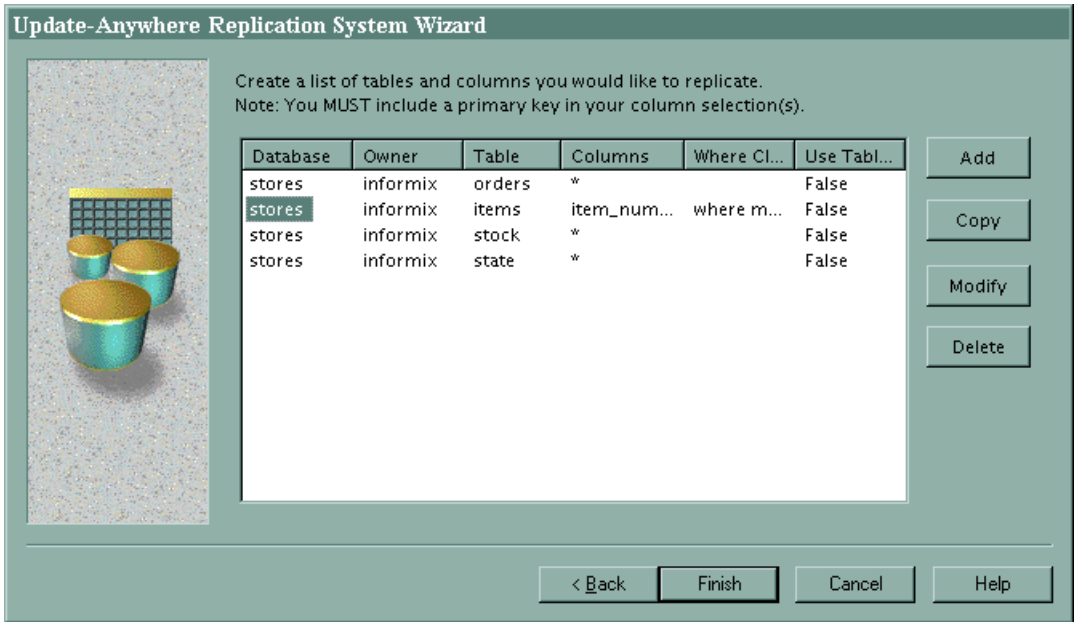
After you choose a naming scheme, the wizard proceeds to ask for attributes of the replicates with pages similar to the wizard pages described in "Defining a Replicate" on page 9-4. Wizard pages ask you to:

- choose a conflict resolution rule.
- choose options for the replicates, such as transaction scope and message format.
- choose a replication frequency.
- choose the database servers that will participate in the replicate.

## Describing the Participants

The next wizard page asks you to choose the tables to replicate. Figure 9-13 shows a list of participant descriptions ready for the generic wizard to create the replicates.

***Figure 9-13***
*Tables to Replicate*



To prepare the information in Figure 9-13, click **Add** for each table that you want to add. Enter the participant attributes, as in Figure 9-14, and click **OK**. Fill in a set of attributes for each table that should be replicated.

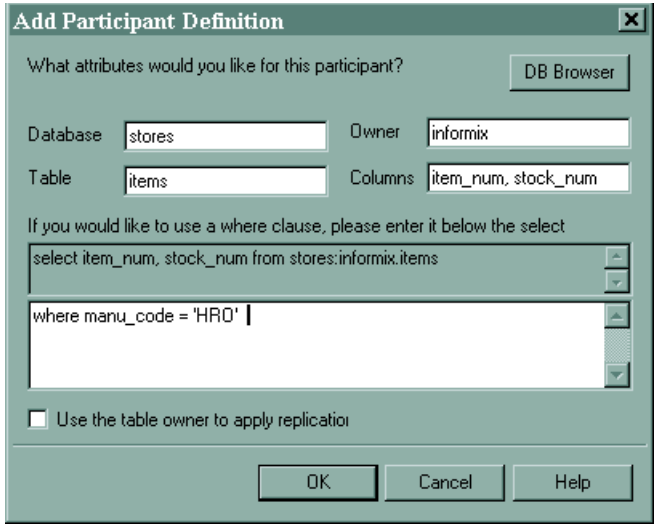



***Figure 9-14***
*Participant Definition*

When you describe all the participants, your display will be similar to Figure 9-13. Click **Finish**.

The Generic Configuration wizard prepares the replicates, as Figure 9-15 shows.

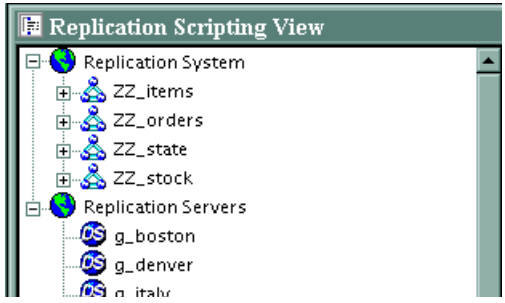


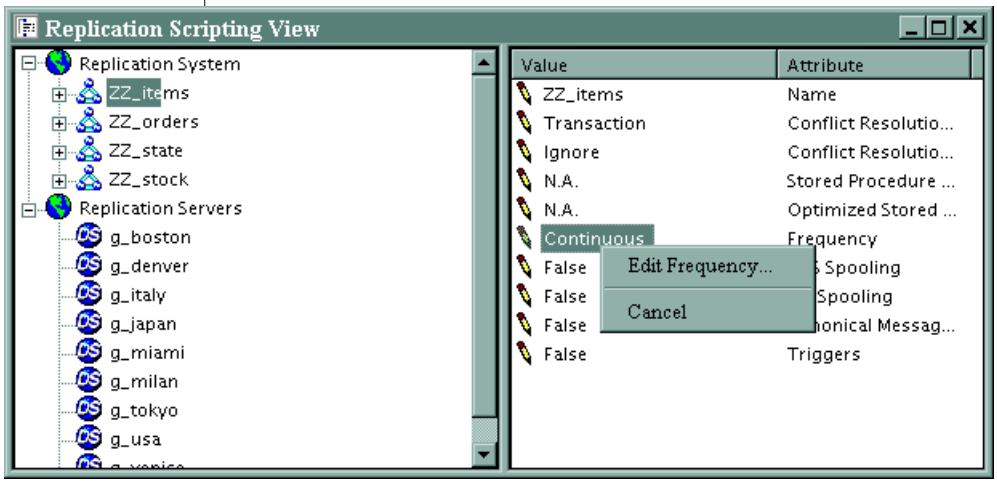

***Figure 9-15***
*Generic Replicates*

## Customizing the Generic Replicates

After you create the generic replicates, you can use the Scripting View to customize the individual replicates. It is usually faster to create generic replicates and then customize as needed, rather than to create each replicate individually.

To customize a replicate, select the replicate and then change its attributes in the right-hand column of the Scripting View. For example, all the **ZZ** replicates have continuous replication, but you want to change the replication of the items table to occur only occasionally. Select **ZZ**_items. Then click with the right button on **Continuous** to get the popup change-frequency menu, as in Figure 9-16.

***Figure 9-16***
*Customizing a Generic Replicate*



Chapter

# 10

# Diagnosing Enterprise Replication


In This Chapter . . . . . 10-3

Aborted-Transaction Spooling . . . . . 10-3
- Preparing to Use ATS. . . . . 10-4
- BYTE and TEXT Information in ATS Files. . . . . 10-8

Row-Information Spooling . . . . . 10-9
- BYTE and TEXT Information in RIS Files . . . . . 10-11

Replication Event Monitor Messages. . . . . 10-14

## In This Chapter

Enterprise Replication provides tools to help you diagnose problems that arise during replications. The aborted-transaction spooling (ATS) and row-information spooling (RIS) files contain information about failed transactions. The Replication Manager logs control events in the Replication Event Monitor. This chapter describes the ATS and RIS files and provides a list of Replication Event Monitor log events.

The **onstat** command, which Appendix F discusses, displays statistics that you can use to diagnose problems.

## Aborted-Transaction Spooling

When ATS is active, all failed replication transactions are recorded in ATS spool files. Each file that ATS produces contains all the information pertinent to a single failed transaction. If a replicated transaction fails for any reason (constraint violation, duplicate, and so on), all the buffers in the replication message that compose the transaction are written to a local operating-system file. You can use the ATS files to identify problems or as input to custom-written utilities that extract or reapply the aborted rows.

In some cases, such as with long transactions, the database server can abort transactions. In these cases, Enterprise Replication does not generate an ATS file.

## Preparing to Use ATS

Failed transactions are not automatically recorded in ATS files. To collect ATS information, perform the following steps:

1. Specify a directory where ATS files will be stored when you define a server for Enterprise Replication. The pathname that specifies the ATS directory can be no longer than 256 bytes.
   - With ERM, set the spooling directories on the Declare Server display that Figure 3-6 on page 3-22 shows.
   - With the CLU, include the **--ats** option with **cdr define server**.

   UNIX

   If you do not specify an ATS directory, Enterprise Replication stores the ATS files in the **/tmp** directory. ♦

   WIN NT

   If you do not specify an ATS directory, Enterprise Replication stores the ATS files in the **\tmp** directory of the drive that contains **%INFORMIXDIR%** ♦
2. Specify that ATS should be active when you define a replicate. In expert mode, you can use ATS for some replicates and not for others.
   - With ERM, activate ATS spooling on the replicate options page (see "Replication Options" on page 3-39).
   - With the APIs, set the **ats_dir** attribute in **cdr_define_repl()**.
   - With the CLU, include the **--ats** option with **cdr define replicate**.

## Naming ATS Spool Files

When ATS is active, each aborted transaction is written to a file in the directory specified in the server definition. The following table provides the naming convention for ATS spool files:

```
ats.target.source.threadId.timestamp.sequence
```

| Name | Description |
|---|---|
| target | The name of the database server receiving this replicate transaction. |
| source | The name of the database server that originated the transaction. |
| threadId | The identifier of the thread that processed this transaction. |
| timestamp | The value of the internal time stamp at the time that this ATS instance was started. |
| sequence | A unique integer. This integer is incremented by ATS each time a transaction is spooled. |

The naming convention ensures that all filenames that ATS generates are unique, and therefore, name collision is unlikely. However, when ATS opens a file for writing, any previous file contents will be overwritten. (ATS does not append to a spool file; if a name collision does occur with an existing file, its original contents will be lost.)

The first three characters in each line of the ATS spool file describe the type of information for the line, as the following table defines.

| Label | Name | Description |
|---|---|---|
| TXH | Transaction Heading | Information from the transaction header that includes the sending server ID and the commit time, the receiving server ID and the received time, and any Enterprise Replication, SQL, or ISAM error information for the transaction. |
| RRH | Replicated Row Heading | Header information from the replicated rows that includes the row number within the transaction, the group ID, the replicate ID (same as replicate group ID if replicate is not part of any replicate group), the database, owner, and table name, and the database operation. |
| RRS | Replicated Row Shadow Columns | Shadow column information from replicated rows that includes the source server ID and the time when the row is updated on the source server. This line is printed only if the replicate is defined with a conflict-resolution rule. |
| RRD | Replicated Row Data | Replicated row data is the list of replicated columns in the same order as in the SELECT statement in the define replicate command. Each column is separated by a '\|' and displayed in ASCII format. When the spooling program encounters severe errors (for example, cannot retrieve replicate id for the replicated row, unable to determine the replicated column's type, size, or length), it displays this row data in hexadecimal format. |

Aborted-transaction spooling only occurs if the entire transaction is aborted. Transactions defined with row scope that have aborted rows but are successfully committed on the target tables are not logged.

All rows that fail conflict resolution *and* have row scope are also written to the RIS file.

The following output is an example of an ATS file for a transaction sent by server **sls4** to server **pur1**. The file is named:

```
ats.pur1.sls4.D_2.960529_23:27:16.6
```

The file contents are:

```
TXH RIS file:/tmp/p1s1/ris.pur1.sls4.D_2.960529_23:27:16.5 has also been
created for this transaction
==========
TXH Source ID:390 / Name:sls4 / CommitTime:96-05-29 23:26:24
TXH Target ID:392 / Name:pur1 / ReceiveTime:96-05-29 23:27:16
TXH Number of rows processed when transaction was aborted:4
TXH All rows in a transaction defined with row scope were rejected
----------
RRH Row:1 / Group Id: 24707077 / Replicate Id: 24707077 / Table: stores@cdr.item
/ DbOp:Update
RRS 390(sls4)|833437578(96/05/29 23:26:18)
RRD 20|VGA Card 439|A|Acer|Chevrolet|VGA Card|439|422.00|538.00|738.00|13
.00|12.00|8.00|Normal Demand Moderate Value Item
----------
RRH Row:2 / Group Id: 24707077 / Replicate Id: 24707077 / Table: stores@cdr.item
/ DbOp:Update
RRS 390(sls4)|833437580(96/05/29 23:26:20)
RRD 10|Digitizer 319|A|Apple|Toyota|Digitizer|319|29.00|36.00|581.00|28.0
0|2.00|2.00|Normal Demand Moderate Value Item
----------
RRH Row:3 / Group Id: 24707080 / Replicate Id: 24707080 / Table:
stores@cdr.customer / DbOp:Update
RRS 390(sls4)|833437582(96/05/29 23:26:22)
RRD 18|Larraine Hustin|2021 South Blvd.|A|8|(399)-2423745|0.00|0.00|0.00|
0.00|83429.00|Untouched|This Customer belongs to the Nation of Kamas
----------
RRH Row:4 / Group Id: 24707077 / Replicate Id: 24707077 / Table: stores@cdr.item
/ DbOp:Update
RRS 390(sls4)|833437583(96/05/29 23:26:23)
RRD 9|Keyboard173|A|ALR|Datsun|Kboard|173|978.00|1399.00|866.00|23.00|1
8.00|12.00|Slow Moving Moderate Value Item
```

## BYTE and TEXT Information in ATS Files

When the information recorded in the ATS file includes BYTE or TEXT data, the replicated row data (RRD) information is reported as the following examples show.

### Example 1

```
<1200, TEXT, PB 877(necromsv) 840338515(96/08/17 20:21:55)>
```

where

- 1200 is the size of the data
- TEXT is the data type (it is either BYTE or TEXT)
- PB is the storage type (PB when the BYTE or TEXT is stored in the tablespace, BB for blobspace storage)
- The next two fields are the server identifier and the time stamp for the column if the conflict-resolution rule is defined for this replicate and the column is stored in a tablespace.

### Example 2

```
<500 (NoChange), TEXT, PB 877(necromsv) 840338478(96/08/17 20:21:18)>
```

where

```
(NoChange)
```

after the 500 indicates that the TEXT data has a size of 500 but the data has not been changed on the source server. Therefore the data is not sent from the source server.

**Example 3**

```
<(Keep local blob),75400, BYTE, PB 877(necromsv) 840338515(96/08/17 20:21:55)>")
```

where

```
(Keep local blob)
```

indicates that the replicated data for this column was not applied on the target table, but instead the local BYTE data was kept. This usually happens when time-stamp conflict resolution is defined and the local column has a time stamp greater than the replicated column.

## Row-Information Spooling

In addition to ATS, Enterprise Replication provides another spooling option called *row-information spooling* (RIS). RIS logs the following types of information:

- Individual aborted row errors
- Replication exceptions (such as when a row is converted by Enterprise Replication from insert to update, or from update to insert, and so on)
- Special stored procedure return codes as defined by the application if a stored procedure is called to resolve a conflict

UNIX

If you do not specify an RIS directory, Enterprise Replication stores the RIS files in the **/tmp** directory. ♦

WIN NT

If you do not specify an ATS directory, Enterprise Replication stores the RIS files in the **\tmp** directory of the drive that contains **%INFORMIXDIR%**. ♦

The pathname for the RIS directory cannot be more than 256 bytes. If the default directory does not exist, Enterprise Replication returns an error.

The RIS row information is written at the time that the data-synchronization component finishes processing the replicated row and can therefore also provide the local row information. (The ATS file is written after the transaction is rolled back, the locks for all the local rows are released and therefore do not include accurate local-row data.) The RIS filename algorithm is analogous to the one that ATS uses, with the **ats** prefix replaced by **ris**. The RIS file that corresponds to the ATS file described in the previous section is, for example:

```
ris.pur1.sls4.D_2.960529_23:27:16.5
```

In addition to the four types of records printed in ATS, the RIS file also includes the following information.

| Label | Name | Description |
|---|---|---|
| LRH | Local-row header | Indicates if the local row is found in the delete table and not in the target table. |
| LRS | Local-row shadow columns | Contains the server id and the time when the row is updated on the target server. This line is printed only if the replicate is defined with a conflict-resolution rule. |
| LRD | Local-row data | Contains the list of replicated columns extracted from the local row and displayed in the same order as the replicated row data. Similar to the replicated row data, each column is separated by a '\|' and written in ASCII format. When the spooling program encounters severe errors (for example, cannot retrieve replicate id for the replicated row, unable to determine the replicated column's type, size, or length), it displays this row data in hexadecimal format. |

## BYTE and TEXT Information in RIS Files

When the information recorded in the RIS file includes BYTE or TEXT data, the RRD information is reported as the following examples show.

### Example 1

```
"<1200, TEXT, PB 877(necromsv) 840338515(96/08/17 20:21:55)>
```

where

- 1200 is the size of the TEXT data
- TEXT is the data type (it is either BYTE or TEXT)
- PB is the storage type (PB for storage in the tablespace, BB for blobspace storage)
- The next two fields are the server identifier and the time stamp for the column if the conflict-resolution rule is defined for this replicate and the column is stored in a blobspace.

### Example 2

```
"<500 (NoChange), TEXT, PB 877(necromsv) 840338478(96/08/17 20:21:18)>
```

where (NoChange) after 500 indicates the TEXT data has size of 500 but the data has not been changed on the source server. Therefore the data is not sent from the source server.

The following output provides some examples of RIS files:

```
TXH Source ID:390 / Name:sls4 / CommitTime:96-05-29 23:26:24
TXH Target ID:392 / Name:pur1 / ReceiveTime:96-05-29 23:27:16
----------
RRH Row:1 / Collection Id: 24707077 / Replicate Id: 24707077 / Table:
stores@cdr.item / DbOp:Update
RRH CDR:14 (Failed conflict resolution rule) / SQL:0 / ISAM:0
LRS pur1(392)|96/05/29 23:27:20(833437640)
LRD 20|VGA Card 439|A|Acer|Chevrolet|VGA Card|439|422.00|538.00|994.00|13
.00|13.00|8.00|Normal Demand Moderate Value Item
RRS sls4(390)|96/05/29 23:26:18(833437578)
RRD 20|VGA Card 439|A|Acer|Chevrolet|VGA Card|439|422.00|538.00|788.00|13
.00|12.00|8.00|Normal Demand Moderate Value Item
----------
RRH Row:2 / Collection Id: 24707077 / Replicate Id: 24707077 / Table:
stores@cdr.item / DbOp:Update
RRH CDR:14 (Failed conflict resolution rule) / SQL:0 / ISAM:0
LRS pur1(392)|96/05/29 23:26:52(833437612)
LRD 10|Digitizer 319|A|Apple|Toyota|Digitizer|319|29.00|36.00|552.00|28.0
0|28.00|2.00|Normal Demand Moderate Value Item
RRS sls4(390)|96/05/29 23:26:20(833437580)
RRD 10|Digitizer 319|A|Apple|Toyota|Digitizer|319|29.00|36.00|581.00|28.0
0|2.00|2.00|Normal Demand Moderate Value Item
----------
RRH Row:3 / Collection Id: 24707080 / Replicate Id: 24707080 / Table:
stores@cdr.customer / DbOp:Update
RRH CDR:14 (Failed conflict resolution rule) / SQL:0 / ISAM:0
LRS ctr0(377)|96/05/29 23:26:35(833437595)
LRD 18|Larraine Hustin|2021 South Blvd.|I|8|(399)-2423745|8469.00|0.00|0.
00|0.00|83429.00|Untouched|This Customer belongs to the Nation of Kamas
RRS sls4(390)|96/05/29 23:26:22(833437582)
RRD 18|Larraine Hustin|2021 South Blvd.|A|8|(399)-2423745|0.00|0.00|0.00|
0.00|83429.00|Untouched|This Customer belongs to the Nation of Kamas
----------
```

```
RRH Row:4 / Collection Id: 24707077 / Replicate Id: 24707077 / Table:
stores@cdr.item / DbOp:Update
RRH CDR:14 (Failed conflict resolution rule) / SQL:0 / ISAM:0
LRS pur1(392)|96/05/29 23:26:48(833437608)
LRD 9|Keyboard 173|A|ALR|Datsun|Keyboard|173|978.00|1399.00|912.00|23.00|
23.00|12.00|Slow Moving Moderate Value Item
RRS sls4(390)|96/05/29 23:26:23(833437583)
RRD 9|Keyboard 173|A|ALR|Datsun|Keyboard|173|978.00|1399.00|866.00|23.00|
18.00|12.00|Slow Moving Moderate Value Item
==========
TXH Transaction aborted

TXH ATS file:/tmp/p1s1/ats.pur1.sls4.D_2.960529_23:27:16.6 has also been created
for this transaction
```

The following RIS output is for a committed transaction with an aborted row and a replication order exception:

```
TXH Source ID:398 / Name:srv3 / CommitTime:96-05-30 08:43:41
TXH Target ID:395 / Name:srv0 / ReceiveTime:96-05-30 08:43:42
----------
RRH Row:2 / Collection Id: 25886728 / Replicate Id: 25886728 / Table: stor
es@cdr.stock / DbOp:Update
RRH CDR:3 (Update exception, row does not exist in target table, converte
d to insert) / SQL:0 / ISAM:0
LRH Local Row in delete:Y
LRS srv0(395)|96/05/30 08:43:39(833471019)
LRD 19|10|A|234.00|0.00|0.00|0.00|Delivery in 3 days|0.00|0.00|0.00|0|0|9
0|1996-05-30 08:42:13
RRS srv3(398)|96/05/30 08:43:41(833471021)
RRD 19|10|A|234.00|327.00|0.00|0.00|Delivery in 3 days|0.00|0.00|0.00|3|2
|19|1996-05-30 08:43:35
----------
RRH Row:1 / Collection Id: 25886737 / Replicate Id: 25886737 / Table: stor
es@cdr.orddetail / DbOp:Delete
RRH CDR:0 / SQL:0 / ISAM:143
LRS srv4(399)|96/05/30 08:43:30(833471010)
LRD 4|5|4|A|5|6.00|4590.00|0.00|0.00|02/04/1998|Issue from Site 2 Only|By
Road|CusCode=15, RecQty=9, OrdQty=33|0.00|0.00|0.00|0.00|4|1|3|1996-05-3
0 12:44:21
RRS srv3(398)|96/05/30 12:43:41(833471021)
RRD 4|5|4|A|5|3.00|2295.00|0.00|0.00|02/06/1998|Please Issue ASAP|By Roa
d|CusCode=15, RecQty=3, OrdQty=39|0.00|0.00|0.00|0.00|4|0|5|1996-05-30 1
2:43:27
==========
TXH Transaction committed
TXH Total number of rows in transaction:5
```

# Replication Event Monitor Messages

The Replication Event Monitor displays messages in the following format:

```
Context : Message.
```

The context part of Context:Message includes an operation, such as define server or suspend group, and the name of the object of the operation. The following lines show three examples of context information:

```
define replicate 'repl_1'
start group 'accounts'
suspend server 'lake'
```

The message part of `Context:Message` consists of a message that describes the problem. The messages are documented in Appendix C.

Three `Context:Message` examples follow:

- `define replicate 'repl_1' : database does not exist`
- `start group 'accounts' : illegal group state change`
- `suspend server 'lake' : insufficient memory to perform operation`

Chapter

# 11

# Command-Line Utility


In This Chapter . . . . . 11-3
List of Commands . . . . . 11-5
Command-Line Abbreviations . . . . . 11-6
Option Abbreviations . . . . . 11-6
Order of the Options . . . . . 11-7
Backslash Use . . . . . 11-8
Connect Option . . . . . 11-8
Participant . . . . . 11-9
Using the O and I Options . . . . . 11-10
Using the P and R Options . . . . . 11-10
Participant Modifier . . . . . 11-11
Restrictions on the SELECT Statement . . . . . 11-11
Database Server Groups . . . . . 11-12
Server Groups on UNIX . . . . . 11-12
Server Groups on Windows NT . . . . . 11-13
Return Codes . . . . . 11-13
Frequency Options . . . . . 11-14
Time of Day . . . . . 11-15
Intervals . . . . . 11-16

Example Using the Command-Line Utility . . . . . 11-16

Command-Line Utility Syntax . . . . . 11-19
cdr change group . . . . . 11-20
cdr change replicate . . . . . 11-22
cdr connect server . . . . . 11-24
cdr define group . . . . . 11-25
cdr define replicate . . . . . 11-27
cdr define server . . . . . 11-33
cdr delete group . . . . . 11-38

cdr delete replicate . . . . . 11-39
cdr delete server . . . . . 11-40
cdr disconnect server . . . . . 11-43
cdr error . . . . . 11-44
cdr list group. . . . . 11-47
cdr list replicate. . . . . 11-48
cdr list server . . . . . 11-50
cdr modify group . . . . . 11-53
cdr modify replicate . . . . . 11-55
cdr modify server . . . . . 11-59
cdr resume group . . . . . 11-61
cdr resume replicate . . . . . 11-62
cdr resume server . . . . . 11-63
cdr start . . . . . 11-64
cdr start group . . . . . 11-66
cdr start replicate . . . . . 11-67
cdr stop . . . . . 11-68
cdr stop group . . . . . 11-70
cdr stop replicate . . . . . 11-71
cdr suspend group. . . . . 11-73
cdr suspend replicate . . . . . 11-75
cdr suspend server. . . . . 11-77

# In This Chapter

The command-line utility (CLU) lets you configure and control Enterprise Replication from the command line on your UNIX or Windows NT operating system.

This chapter has the following sections:

- Definitions
  - List of commands
  - Command-line abbreviations
  - Option abbreviations
  - Backslash use
  - Connect clause
  - Participant
  - Participant modifier
  - Server group
  - Return codes
  - Time formats
- Example of a command-line script
- Descriptions of each individual command-line command
- Definitions

This section defines the terminology and conventions used in the descriptions of the CLU.

The CLU has many variations, such as **cdr define group** and **cdr resume replication**. Although technically Enterprise Replication has only *one* command-line utility, **cdr**, this chapter treats each variation of the CLU as if it were a separate command.

Each command follows the same approximate format, with the following components:

- Command and its variation

  The command specifies the action that should be taken.

- Options

  The options modify the action of the command. Each option starts with a minus (-) or a double-minus (--).

- Target

  The target specifies the Enterprise Replication object that should be acted upon.

- Other objects

  Other objects specify objects that are affected by the change to the target.

If you enter an incorrect **cdr** command at the command-line prompt, the database server returns a usage message that summarizes the **cdr** commands. For a more detailed usage message, enter **cdr *variation* -h**. For example, **cdr list server -h**.

The following table shows a few examples of the **cdr** commands.

| Command | Options | Target | Other Objects |
|---|---|---|---|
| **cdr define group** | --connect ohio | accounts | replicate1 replicate2 |
| **cdr modify group** | --sequential | accounts | |
| **cdr sta gro** | -c ohio | accounts | g_server1 g_server2 |
| **cdr def ser** | -I -L -S | corporate | store1 |
| **cdr list replicate** | | | replicate1 |

***Important:*** *You must be user **informix** to execute any of the CLU commands except the **cdr** list options.*

## List of Commands

The following table shows the commands and the page where the command is documented.


| Command | Page |
|---|---|
| **cdr change group** | 11-20 |
| **cdr change replicate** | 11-22 |
| **cdr connect server** | 11-24 |
| **cdr define group** | 11-25 |
| **cdr define replicate** | 11-27 |
| **cdr define server** | 11-33 |
| **cdr delete group** | 11-38 |
| **cdr delete replicate** | 11-39 |
| **cdr delete server** | 11-40 |
| **cdr disconnect server** | 11-43 |
| **cdr error** | 11-44 |
| **cdr list group** | 11-47 |
| **cdr list replicate** | 11-48 |
| **cdr list server** | 11-50 |
| **cdr modify group** | 11-53 |
| **cdr modify replicate** | 11-55 |
| **cdr modify server** | 11-59 |
| **cdr resume group** | 11-61 |
| **cdr resume replicate** | 11-62 |
| **cdr resume server** | 11-63 |
| **cdr start** | 11-64 |


(1 of 2)

| Command | Page |
|---|---|
| **cdr start group** | 11-66 |
| **cdr start replicate** | 11-67 |
| **cdr stop** | 11-68 |
| **cdr stop group** | 11-70 |
| **cdr stop replicate** | 11-71 |
| **cdr suspend group** | 11-73 |
| **cdr suspend replicate** | 11-75 |
| **cdr suspend server** | 11-77 |

(2 of 2)

## Command-Line Abbreviations

Each of the words that make up a **cdr** command variation can be abbreviated to three or more characters. For example, the following commands are all equivalent:

```
cdr define replicate
cdr define repl
cdr def rep
```

## Option Abbreviations

Each option for a command has a long form and a short form. You can use either form, and you can mix long and short forms within a single command. For example, using long forms you can write:

```
cdr define server --connect=ohio --idle=500 --send=dbs1 \
--recv=dbs2 --ats=/cdr/ats --initial utah
```

Using short forms, you can write the previous example as follows:

```
cdr def ser -c ohio -i 500 -s dbs1 -r dbs2 -A /cdr/ats -I utah
```

The long form is always preceded by a double minus (--). Most (but not all) long forms require an equals sign (=) between the option and its argument. The short form is preceded by a single minus (-) and is the first letter of the long form. The short form never requires an equals sign. However, sometimes the short form is capitalized and sometimes not. To find the correct syntax for the short form, check the table that accompanies each command variation.

***Tip:*** *Informix suggests that you use the long forms of options when you are preparing a script that might be used by someone other than yourself.*

With the exception of the keyword `transaction`, all keywords (or letter combinations) that modify the command options must be written as shown in the syntax diagrams. For example, in the "Conflict Options" on page 11-28, the option (`conflict`) can be abbreviated, but the keyword `ignore` cannot be abbreviated. Both of the following forms are correct:

```
--conflict=ignore
-C ignore
```

## Order of the Options

You can specify the options of the CLU commands in any order. Some of the syntax diagrams in this chapter show the options in a specific order because it makes the diagram easier to read. You can choose any order that you want for the options.

You should not repeat an option. The following fragment is incorrect because `-c` appears twice. In most cases, the CLU will catch this inconsistency and flag it as an error. However, if no error is given, the database server uses the last instance of the option. In the following example, the database server uses the value `-c=utah`.

```
-c=ohio -i=500 -c=utah
```

***Tip:*** *Informix suggests that, for ease of maintenance, you always use the same order for your options.*

## Backslash Use

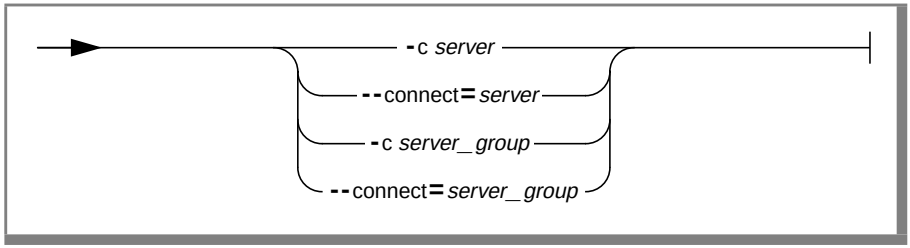
Many CLU commands become quite long when all the options are included. The examples in this chapter use a backslash (\) to indicate that a command continues on the next line. The following two commands are equivalent. The first command is too long to fit on a single line, so it is continued on the next line. The second example, which uses short forms for the options, fits on one line.

```
cdr define server --connect=ohio --idle=500 --send=dbs1 \
--recv=dbs2 --ats=/cdr/ats

cdr define server -c=ohio -i=500 -s=dbs1 -r=dbs2 -A=/cdr/ats
```

Check the documentation of your operating system to find out how to manage long lines at your command prompt.

## Connect Option

The **connect** option causes the command to use the global catalog that is on the specified server. If this option is not invoked, the connection defaults to the database server that the **INFORMIXSERVER** environment variable specifies.




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *server* | Name of the database server to connect to | Must be the name of a database server that is registered with Enterprise Replication. | Identifiers, p. 8-8 |
| *server_group* | Name of the server group that includes the database server to connect to | Must be the name of an existing database server group. | Identifiers, p. 8-8 |

You must use the **connect** option when you add a database server to your replication environment with the **cdr define server** command.

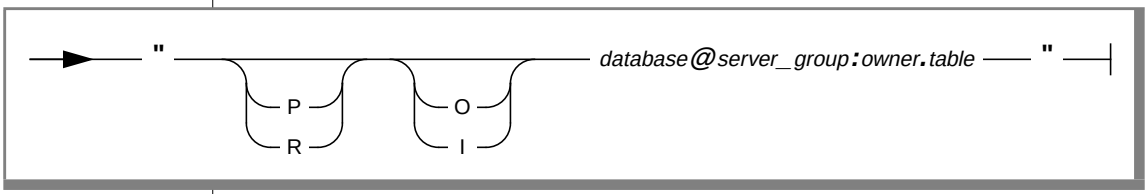
You might use the **connect** option if the database server to which you would normally attach is unavailable or, in some limited cases, to improve efficiency.

For more information about the global catalog, refer to "Catalog" on page 2-14.

## Participant

The participant identifies a table that should be replicated. You must include the server group, database, table owner, and table name. The participant must be enclosed in quote marks. This definition of a participant is used in the following CLU options:

- **cdr define replicate**
- **cdr modify replicate**
- **cdr change replicate**




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *database* | Name of the database that includes the table to be replicated | The database server must be registered with Enterprise Replication. | Identifiers, p. 8-8 |
| *owner* | User ID of the owner of the table to be replicated | None. | Identifiers, p. 8-8 |
| *server_group* | Name of the server group that includes the server to connect to | Must be the name of an existing server group. | Identifiers, p. 8-8 |
| *table* | Name of the table to be replicated | The table must be an actual table. It cannot be a synonym or a view. | Identifiers, p. 8-8 |

### Using the O and I Options

The O (owner) option causes permission checks for *owner* to be applied to the operation (such as insert or update) that is to be replicated and to all actions fired by any triggers. When the **O** option is omitted, all operations are performed with the privileges of user **informix**.

UNIX

If a trigger requires any system-level commands (as specified using the **system()** command in an SPL statement), the system-level commands are executed as the table owner if the participant includes the **O** option. ♦

WIN NT

If a trigger requires any system-level commands, the system-level commands are executed as a less privileged user because on Windows NT you cannot impersonate another user without having the password, whether or not the participant includes the **O** option. ♦

Use the **I** option to disable the table-owner option.

### Using the P and R Options

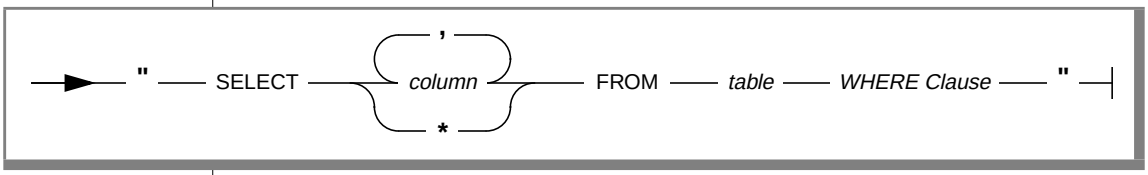
You can specify the *mode* of replication by using the **P** (primary) and **R** (read only, or target) options with the participant. If you do not specify either **P** and / or **R** for the participants, then the replicate is an update-anywhere replicate. For example, in the following definitions, replicate **Rone** is update-anywhere, while in replicate **Rtwo**, **serv2** is the primary and **serv1** is read-only.

```
cdr define repl-c serv1 -C timestamp -S tran Rone \
"db@serv1:owner.table" "select * from table" \
"db@serv2:owner.table" "select * from table"

cdr define repl-c serv1 -C ignore -S tran Rtwo \
"R db@serv1:owner.table" "select * from table" \
"P db@serv2:owner.table" "select * from table"
```

For a discussion of primary and target servers, refer to "Primary-Target Replication" on page 2-15.

## Participant Modifier

The participant *modifier* is a restricted SELECT statement. The participant modifier specifies the rows and columns that will be replicated. The participant modifier must be enclosed in quote marks..




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *column* | Name of a column in the table specified by the participant | The column must exist. | Identifiers, p. 8-8 |
| *table* | The table specified by the participant | Must be the table name only. You cannot specify an owner or a database server. You cannot specify a synonym or a view. | Identifiers, p. 8-8 |
| *WHERE Clause* | Clause that specifies a subset of table rows to be replicated | | WHERE clause syntax, refer to *Informix Guide to SQL: Syntax* |

### *Restrictions on the SELECT Statement*

The following restrictions apply to a SELECT statement that is used as a participant modifier:

- The FROM clause of the SELECT statement can reference only a single table.
- The statement cannot include a join or a subquery.
- The table in the FROM clause must be the table specified by the participant.
- The columns selected must include the primary key.
- The columns selected cannot include smart large objects. (The columns selected *can* include TEXT and BYTE data.)

- The columns selected can include only limited user-defined types. For information about using user-defined data types, refer to "Replicating User-Defined Data Types" on page 5-15.
- The statement cannot contain user-defined routines.

For detailed information about the SELECT statement, refer to the *Informix Guide to SQL: Syntax*.

## Database Server Groups

Except in the **Connect** option, the CLU uses the name of the *database server group* instead of the more familiar *database server* name (that is, the name specified in the **INFORMIXSERVER** environment variable) for all references to database servers. This manual also refers to a database server group as a *server group*.

Typically, a server group includes only one database server. However, if the computer has multiple network protocols and / or network interface cards, the server group includes all aliases for the database server. Enterprise Replication treats the server group as one object, whether it includes one or several database server names.

If you use both DBSERVERNAME and DBSERVERALIASES, DBSERVERNAME should refer to the network connection and not to a shared-memory connection. For information about database server aliases, refer to the *Administrator's Guide for Informix Dynamic Server 2000*.

UNIX

### Server Groups on UNIX

On UNIX, a server group is defined in the **sqlhosts** file. In the following example, line 2 describes the database server **aserver**. Lines 3 and 4 define two more database servers that are aliases for **aserver**. Line 1 defines a server group, **g_aserver**, that includes **aserver**, **bserver**, and **aserver_shm**.

```
1  g_aserver     group       -          -          i=27
2  aserver       ontlitcp    myhost     svcname2   g=g_aserver
3  bserver       ontlitcp    myhost     svcname9   g=g_aserver
4  aserver_shm   onipcshm    myhost     xyz        g=g_aserver
```

All the database server names (dbservername and dbserver aliases) must be grouped together after the line that defines the database server group.

***Tip:*** *If you are using UNIX* ***exclusively****, you can assign the same name to the database server group and the database server to avoid having to remember when to use the name of the database server group or the name of the database server. You* ***cannot*** *do this if you use ERM or if any of your database servers reside on Windows NT computers.*

WIN NT

### Server Groups on Windows NT

With Dynamic Server, Version 7.31, use the IECC Add Database Servers wizard to prepare the SQLHOSTS connectivity information.

With Dynamic Server, Version 9.2, refer to the documentation notes described in "Documentation Notes, Release Notes, Machine Notes" on page 15 of the Introduction for information about preparing the SQLHOSTS connectivity information. ♦

## Return Codes

If the command has an error, the database server returns an error message and a return-code value. The message briefly describes the error. The return-code value corresponds to the return-code values listed in Appendix C.

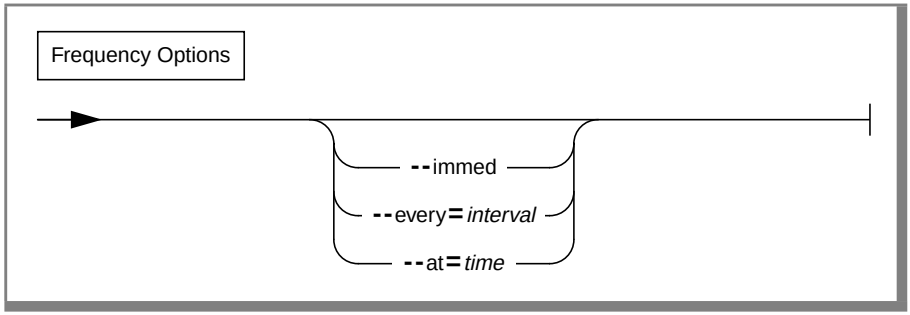
For example, the following command is incomplete because it does not include a conflict-resolution rule. The (58) is the error number as listed in Appendix C.

```
> cdr def rep r12 \
"host1@g_svr1:informix.customer" "select * from customer"
"host2@g_svr2:informix.customer" "select * from customer"

command failed -- No conflict resolution rule specified  (58)
```

## Frequency Options

The following CLU options allow you to specify the interval between replications or the time of day when an action should occur. If you do not specify a time, the action occurs immediately.

- **cdr define group**
- **cdr define replicate**
- **cdr modify group**
- **cdr modify replicate**




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *interval* | Time interval for replication | The smallest interval is minutes. | Intervals, p. 11-16 |
| *time* | Specific time for replication | Time is given with respect to a 24-hour clock. | Time of Day, p. 11-15 |

The time options have the meanings that the following table shows.

| Time Options Long Form | Short Form | Meaning |
|---|---|---|
| --immed | -i | Action occurs immediately. |
| --every | -e | Action occurs immediately and repeats at the frequency specified by interval. |
| --at | -a | Action occurs at the specified day and time. |

### Time of Day

The time of day is always given with respect to a 24-hour clock. For example, 19:30 is 7:30 P.M. Times are always given with respect to the local time, unless the **$TZ** environment variable is set. However, Enterprise Replication stores times in the global catalog in Greenwich mean time (GMT).

The `-a time` option lets you specify the day on which an action should occur. The string *time* can have the following formats:

- Day of week

  To specify a specific day of the week, give either the long or short form of the day, followed by a period and then the time. For example, `--atime Sunday.18:40` or `-a Sun.18:40`, specifies that the action should occur every Sunday at 6:40 P.M.

  The day of the week is given in the locale of the client. For example, with a French locale, you might have `--atime Lunedi.3:30` or `-a Lun.3:30`. The time and day are in the time zone of the server.

- Day of month

  To specify a specific date in the month, give the date, followed by a period, and then the time. For example, `1.3:00` specifies that the action should occur at 3 A.M. on the first day of every month.

  The special character `L` represents the last day of the month. For example, `L.17:00` is 5 P.M. on the last day of the month.

- Daily

  To specify that replication should occur each day, give only the time. For example, 4:40 specifies that the action should occur every day at 4:40 A.M.

### Intervals

The `-e length` option lets you specify the interval between actions. The *length* of time between replications can be either of the following formats:

- The number of minutes

  To specify the number of minutes, specify an integer value. For example, `-e 60` indicates 60 minutes between replications.

- The number of hours and minutes

  To specify hours and minutes, give the number of hours, followed by a colon, and then the number of minutes. For example, `-e 5:45` indicates 5 hours and 45 minutes between replications.

## Example Using the Command-Line Utility

This section contains a simple example of replication that uses the CLU.

### Prepare the Environment

To run this replication example, you need two Informix database servers. Each database server must be in a separate database server group. You *can* run two database server instances on one computer, but it is more realistic to use database servers on different computers. The example uses the following environment:

- Two computers are hosts for the database servers: **urban** and **rural**. All commands in this example, except for creation of the sample database **lake_db**, are issued by a person working on **urban**.
- The database servers are named **hill** and **lake**.
- The database servers **hill** and **lake** are members of the database server groups **g_hill** and **g_lake**, respectively.

- The databases for this example are identical **stores_demo** databases with logging:

  On the **hill** host (urban), create the **hill_db** database:

  ```
  urban> dbaccessdemo hill_db -log
  ```

  On the **lake** host (rural), create the **lake_db** database:

  ```
  rural> dbaccessdemo lake_db -log
  ```

## Declare the Database Servers for Replication

Before replicating data, you must declare the database servers to Enterprise Replication, as follows:

1. Decide where the send queue should go.

   The send queue location is one of the options of **cdr define serve**r. If you do not specify a location for the send queue, it defaults to root dbspace. This example uses the default.

2. Declare the **hill** database server to Enterprise Replication.

   ```
   urban> cdr define server --init g_hill
   ```

   When you define a server for Enterprise Replication, you *must* use the name of the server group rather than the database server name.

3. Verify that the definition succeeded. You can use one of the following actions:

   - Use the CLU utility **cdr list server** to list existing Enterprise Replication servers.
   - The first Enterprise Replication definition on a database server creates a **syscdr** database. Use DB-Access and choose **Database→Select** to verify that **syscdr** is present.

     The **syscdr** database is not designed to provide information about Enterprise Replication to users; it holds information that the database server uses to manage Enterprise Replication. Information about the status of Enterprise Replication is in the **sysmaster** database tables that Appendix B, "SMI Tables for Enterprise Replication" discusses.

4. Declare the second database server, **lake**, to Enterprise Replication.

```
urban> cdr def ser -c lake --init --sync g_hill g_lake
```

   The `--connect` option allows you to declare **g_lake** to Enterprise Replication (and thus create the **syscdr** database) on the database server **lake** while working on the **urban** computer.

   The `--sync` option instructs the command to use the already-defined server (**g_hill**) as a pattern for the new definition.

5. Verify that the second definition succeeded. You can use one of the following actions:

   - Use the CLU option **cdr list server**.
   - Use DB-Access to check the Enterprise Replication servers in the **sysmaster** database. The following two statements should show the same results:

```
SELECT servid, servname FROM sysmaster@hill:syscdrs
SELECT servid, servname FROM sysmaster@lake:syscdrs
```

## Define Replicates

The replicate definition has four basic parts: options, the replicate name, and two participants.

1. Define a replicate that records all changes to the **customer** table:

```
urban> cdr define repl -C ignore repl1 \
"hill_db@g_hill:informix.customer" \
"select * from customer" \
"lake_db@g_lake:informix.customer" \
"select * from customer"
```

2. Verify that the definition succeeded. You can use one of the following actions:

   - Use the CLU option **cdr list replicate**.
   - Use DB-Access to check the replicate definitions:

```
SELECT replname, replstate FROM sysmaster:syscerrepl
```

## Start Replication

You have now *described* the replication environment, but no replication occurs until you instruct Enterprise Replication to start replicating. You can start and stop each replicate individually.

1. Start replication for the replicate that you just defined, **repl1**:

```
urban> cdr start repl -c hill repl1
```

2. Verify that the replication is active. Execute **cdr list replicate** again and verify that the STATE column has changed from INACTI to ACTIVE.

   At this point replication is active, but no database activity has occurred.

3. Use DB-Access to make a change to the **hill** database:

```
INSERT INTO customer VALUES (0,'alice','adams',
'Aardvark', 121 First Street','','Atlanta','IN',
'47904', '317-463-5555')
```

4. Use DB-Access to verify that the change to **hill_db** database is reflected on the **lake_db** database:

```
SELECT * FROM lake_db@lake:customer
WHERE fname = 'alice'
```

# Command-Line Utility Syntax

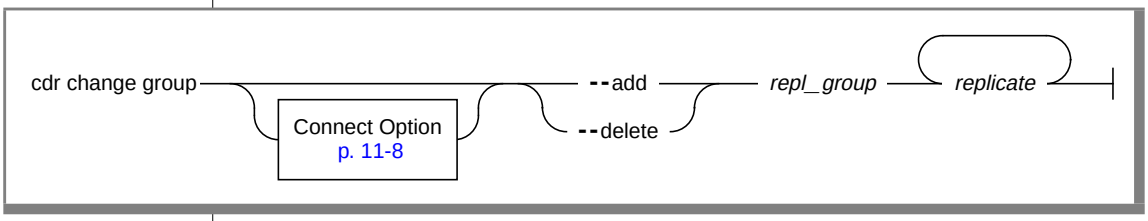
The next sections show the syntax for all the variations of the **cdr** CLU.

The error messages that Enterprise Replication returns when a command fails are documented in Appendix C.

# cdr change group

The **cdr change group** command changes a replicate group by adding or deleting replicates.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_group* | Name of the replicate group to change | The replicate group must exist. | Identifiers, p. 8-8 |
| *replicate* | Name of the replicate(s) to add to or delete from the group | The replicate(s) must exist. | Identifiers, p. 8-8 |

The options in **cdr change group** have the meanings that the following table shows.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --add | -a | Add replicate(s) to a group |
| --delete | -d | Remove replicate(s) from a group |

## Usage

Use this command to add replicates to a group or to remove replicates from a group.

You cannot remove replicates from a group when the group is suspended. Also, you cannot add a suspended replicate to a group, even if the group is also suspended.

## Examples

The following example adds the replicates **house** and **barn** to group **newprod**:

```
cdr change group -add newprod house barn
```

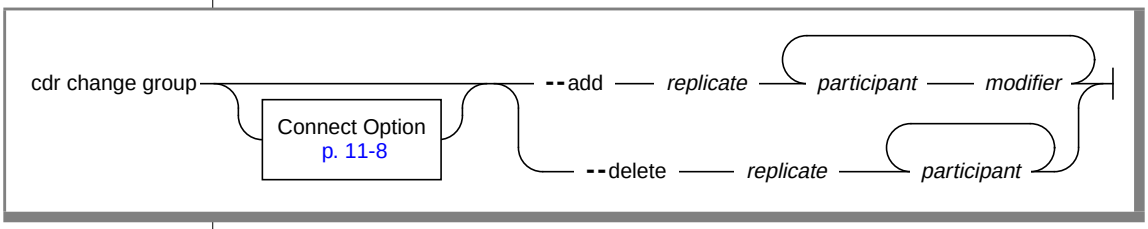
The following example removes the replicates **teepee** and **wigwam** from group **favorites**:

```
cdr cha gro -del favorites teepee wigwam
```

# cdr change replicate

The **cdr change replicate** command allows you to modify an existing replicate by adding or deleting one or more participants.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *modifier* | Specifies the rows and columns to replicate | See "Restrictions on the SELECT Statement" on page 11-11. | Participant Modifier, p. 11-11 |
| *participant* | Specifies the database server and table for replication | The participant must exist. | Participant, p. 11-9 |
| *replicate* | Name of the replicate to change | The replicate must exist. | Identifiers, p. 8-8 |

The options in **cdr change replicate** have the meanings that the following table shows.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --add | -a | Add participant(s) to a replicate |
| --delete | -d | Remove participant(s) from a replicate |

## Usage

Use this command to add or delete a participant from a replicate. You can define a replicate that has zero or one participants but to be useful, a replicate must have at least two participants.

To add or delete a participant using ERM, refer to "Managing Replicates and Participants" on page 8-13.

## Examples

The following example adds two participants to the replicate named **repl_1**: **db1@server1:antonio.table** with the modifier `select * from table1`, and **db2@server2:carlo.table2** with the modifier `select * from table2`:

```
cdr change repl -a repl_1 \
    "db1@server1:antonio.table1" "select * from table1" \
    "db2@server2:carlo.table2" "select * from table2"
```

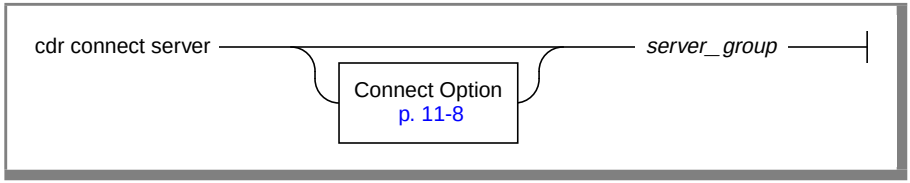
The following example removes the same two participants from replicate **repl_1**:

```
cdr change repl -d repl_1 \
    "db1@server1:antonio.table1" \
    "db2@server2:carlo.table2"
```

# cdr connect server

The **cdr connect server** command re-establishes a connection to a database server that has been disconnected with a **cdr disconnect server** command.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *server_group* | Name of server group to resume | The server group must be defined for replication and have been disconnected. | Server Group, p. 11-12 |

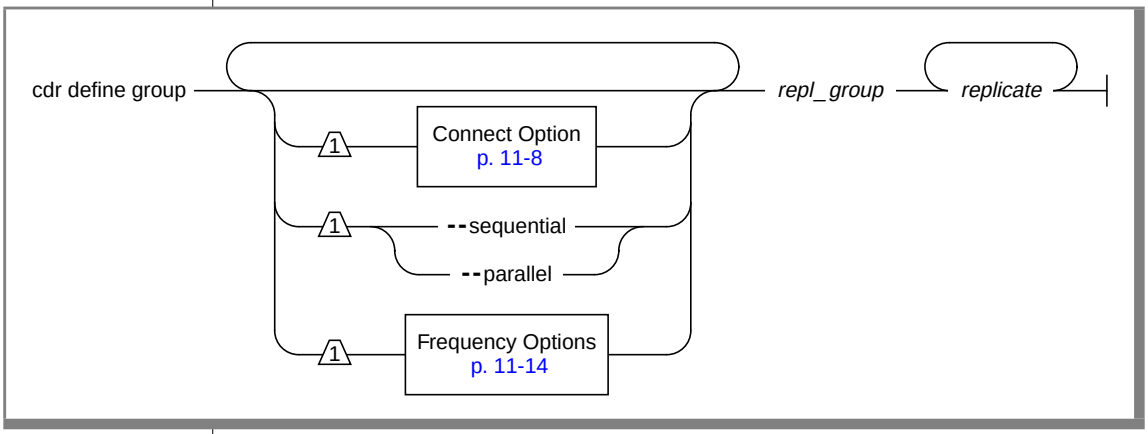
## Usage

The **cdr connect server** command re-establishes a connection to the server *server_group*.

# cdr define group

The **cdr define group** command defines a replicate group.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_group* | Name of replicate group to create | The name must be unique and cannot be the same as a replicate name. | Identifiers, p. 8-8 |
| *replicate* | Name of a replicate to be included in the group | The replicate must exist. | Identifiers, p. 8-8 |

The options in **cdr define group** have the meanings that the following table shows.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --parallel | -p | Process all groups in parallel. |
| --sequential | -s | Process all groups sequentially. |

You can specify parallel options only when replicates in the group have disjoint data domains. If neither the parallel nor sequential option is set, the default is sequential processing.

## Usage

The **cdr define group** command defines a replicate group. Any valid replicate can be defined as part of a replicate group. However, each replicate in the replicate group must meet the following restrictions:

- All the replicates must be defined over the same set of database servers.
- All the replicates must be in the same replication state.

  For information about replication states, refer to "Changing Replicate States" on page 8-14.
- A replicate can be in only one group at a time.

Once a replicate is defined as part of a replicate group, control and configuration is only allowed with replicate-group control commands. You cannot control a replicate individually until either the replicate is removed from the replicate group or the replicate group is deleted.

For a discussion of parallel versus sequential processing, refer to "Replicate Group" on page 2-12.

## Example

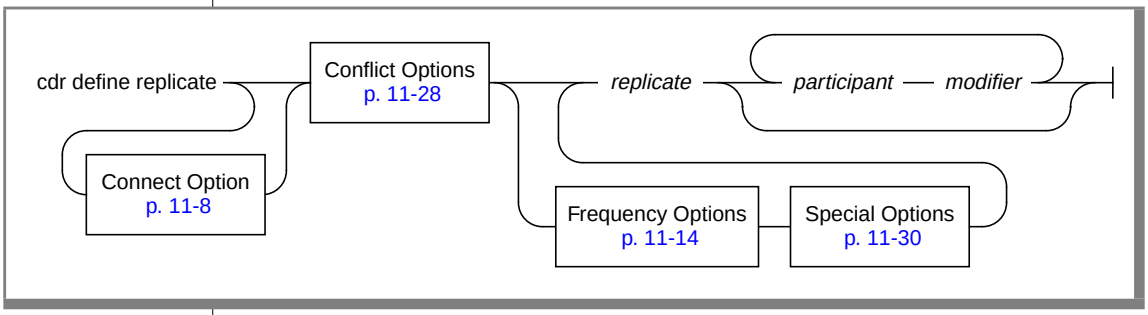
The following example connects to the default server and defines the replicate group **accounts** with replicate participants **repl1**, **repl2**, and **repl3**:

```
cdr define group accounts repl1 repl2 repl3
```

# cdr define replicate

The **cdr define replicate** command defines a replicate in the global catalog.

## Syntax



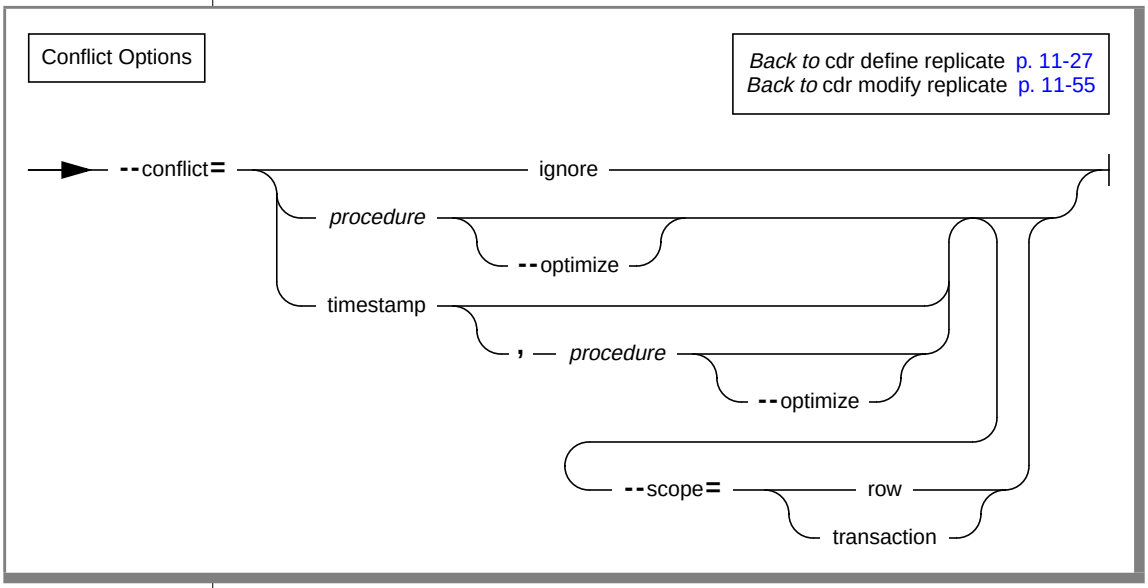

| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *modifier* | Specifies the rows and columns to replicate | See "Restrictions on the SELECT Statement" on page 11-11. | Participant Modifier, p. 11-11 |
| *participant* | Name of a participant in the replication | The participant must exist. | Participant, p. 11-9 |
| *replicate* | Name of the new replicate | The replicate name must be unique. | Identifiers, p. 8-8 |

## Usage

To be useful, a replicate must include at least two participants. You can define a replicate that has only one participant, but before you can use that replicate, you must use **cdr modify replicate** to add more participants.

## Conflict Options

The **conflict** options specify how Enterprise Replication should resolve conflicts among data that arrive at the database server.



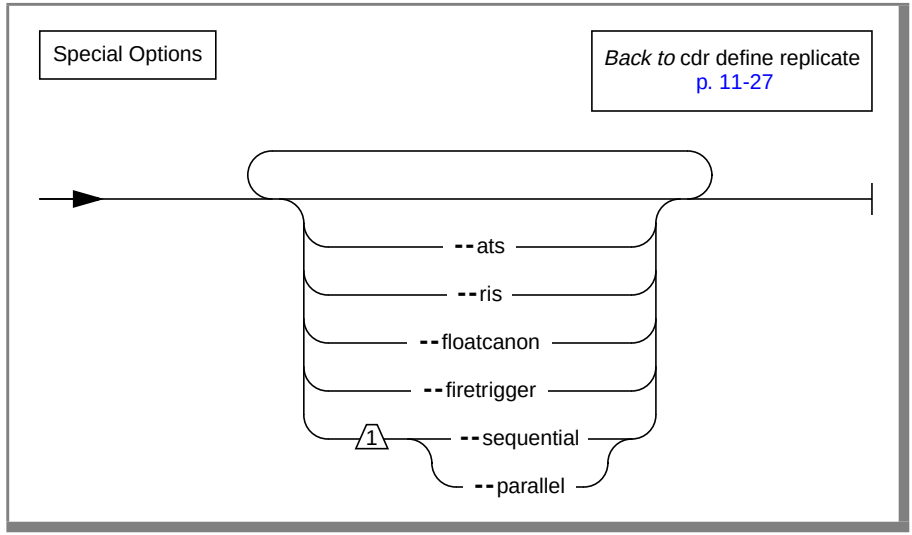

| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *procedure* | Stored procedure for conflict resolution | The procedure must exist. | Identifiers, p. 8-8 |

For a discussion of conflict resolution, refer to "Evaluating Data for Replication" on page 7-4.

The **conflict** options have the meanings shown in the following table.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --conflict | -C | Specifies the rule that will be used for conflict resolution. |
| --optimize | -O | When a conflict occurs and the row to be replicated meets the following conditions, the replicated row is accepted and the stored procedure is not called:<br>■ The row is from the same database server that last updated the local row on the target table.<br>■ The row has a time stamp greater than or equal to the local row.<br>When this option is not present, the stored procedure defined for the replicate is always called when conflict is detected. |
| --scope | -S | Specifies the scope for conflict resolution. If scope is not specified, the default scope is transaction.<br>When specifying the scope, you can abbreviate *transaction* to fewer letters. |

## Special Options




The special options for **cdr define replicate** have the meanings that the following table shows.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --ats | -A | Activate aborted-transaction spooling for replicate transactions that fail to be applied to the target database. For more information, see Chapter 10, "Diagnosing Enterprise Replication." |
| -ris | -R | Activates row-information spooling for replicate row data that fails conflict resolution or encounters replication order problems. For more information, see Chapter 10, "Diagnosing Enterprise Replication." |
| --floatcanon | -F | When floating-point numbers are replicated, transfer the numbers in canonical (machine-independent) format. This format is portable between operating systems but can lose accuracy. |

(1 of 2)

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --firetrigger | -T | The rows that this replicate inserts should fire triggers at the destination. |
| --parallel | -p | Process members in parallel. The default is serial processing. |
| --sequential | -s | Process members sequentially. |

(2 of 2)

## Examples

The following example illustrates the use of **cdr define replicate**:

```
1 cdr define repl --conflict=timestamp,sp1 \
2 --scope=tran --ats newrepl \
3 "db1@iowa:antonio.table1" "select * from table1" \
4 "db2@utah:carlo.table2" "select * from table2"
```

Line 1 of the example specifies a primary conflict-resolution rule of *time stamp*. If the primary rule fails, the stored procedure **sp1** will be invoked to resolve the conflict. Because no database server is specified here (or on any later line), the command connects to the database server named in the **INFORMIXSERVER** environment variable.

Line 2 specifies that the replicate has a transaction scope for conflict-resolution scope and enables aborted-transaction spooling. The final item specifies the name of the replicate, **newrepl**.

Line 3 defines the first participant, "`db1@iowa:antonio.table1`", with the select statement "`select * from table1`".

Line 4 defines a second participant, "`db2@utah:carlo.table2`", with the select statement "`select * from table2`".

The next example is the same as the preceding example with the following exceptions:

Line 1 instructs Enterprise Replication to use the global catalog on database server **ohio**. Line 2 also specifies that the data should be replicated every five hours.

```
1 cdr def repl -c ohio -C timestamp,sp1 \
2 -S tran -A -e 5:00 newrepl \
3 “db1@iowa:antonio.table1” “select * from table1” \
4 “db2@utah:carlo.table2” “select * from table2”
```

The following example is the same as the preceding example except data will be replicated daily at 1:00 A.M.:

```
cdr define repl -c ohio -C timestamp,sp1 \
-s tran -A -a 1:00 replname \
“db1@iowa:antonio.table1” “select * from table1” \
“db2@utah:carlo.table2” “select * from table2”
```

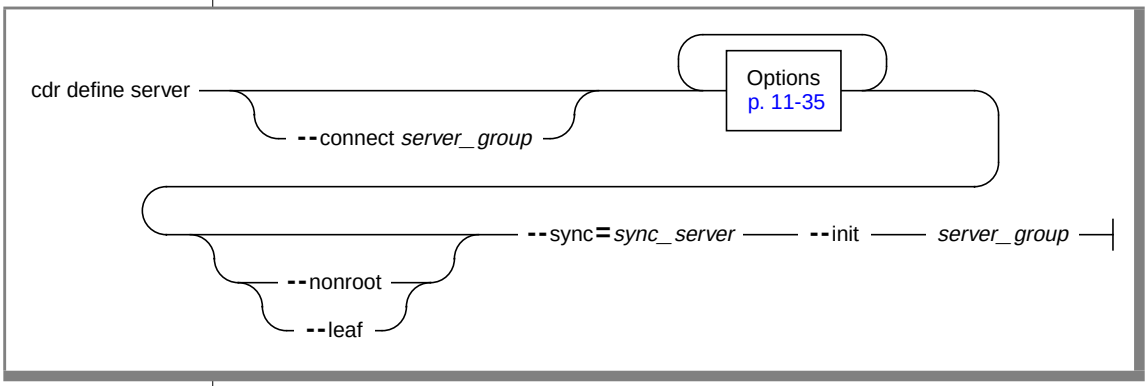
The following example is the same as the preceding example except data will be replicated on the last day of every month at 5:00 A.M.:

```
cdr define repl -c ohio -C timestamp,sp1 \
-S tran -A -a L.5:00 replname \
“db1@iowa:antonio.table1” “select * from table” \
“db2@utah:carlo.table2” “select * from table2”
```

# cdr define server

The **cdr define server** command defines a database server group and all its members (that is, all database servers that are members of the database server group) for Enterprise Replication.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *server_group* | Name of a server group to declare for Enterprise Replication | Must be the name of an existing server group. | Database Server Group, p. 11-12 |
| *sync_server* | Name of server to use for synchronization | Must be a server that is registered with Enterprise Replication. The server must be on-line. | Identifiers, p. 8-8 |

The long forms shown in the syntax diagram have the meanings and abbreviations shown in the following table.

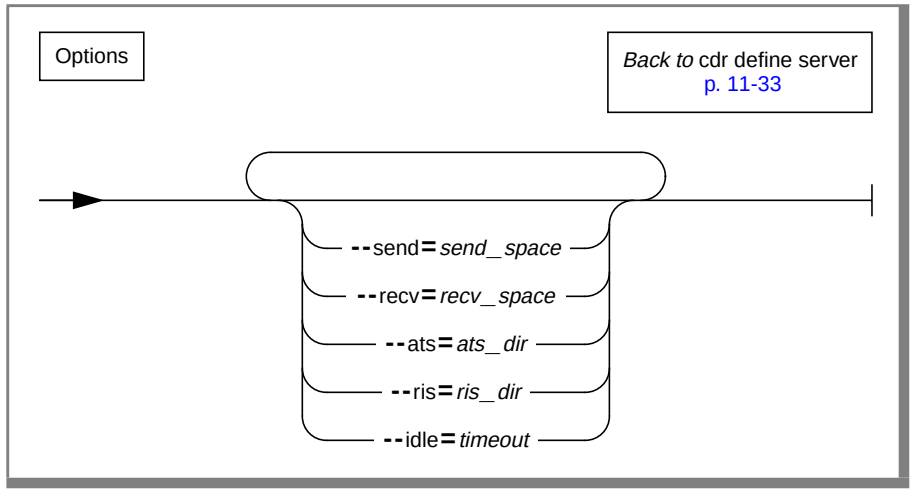
| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --connect | -c | Connects to the database server that is being defined. If this option is omitted, INFORMIXSERVER must be set to *server_group*. |
| --init | -I | Adds *server_group* to the replication system.<br>The *server_group* must be the same as the connection server. |
| --leaf | -L | Defines the server as a leaf server. If neither leaf nor nonroot is specified, the server is defined as a root server. |
| --nonroot | -N | Defines the server as a nonroot server. If neither leaf nor nonroot is specified, the server is defined as a root server. |
| --sync= | -S | Uses the global catalog on *sync_server* as the template for the global catalog on the new replication server, *server_group*. |

## Usage

The **cdr define server** command enters a database server from the *server_group* into the Enterprise Replication global catalog.

## Options

The options allow you to modify the default behavior of **cdr define server**.




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *ats_dir* | Name of the directory for aborted-transaction spooling | Must be a full pathname. The path for the directory can be no longer than 256 bytes. | Follows naming conventions on your operating system. |
| *recv_space* | Name of the dbspace for the receive queue | The dbspace must already exist. The path for the receive queue can be no longer than 128 bytes. The recv_space cannot be a temporary dbspace. | Identifiers, p. 8-8 |
| *ris_dir* | Name of the directory for row-information spooling | Must be a full pathname. The path for the directory can be no longer than 256 characters. | Follows naming conventions on your operating system. |
| *send_space* | Name of the dbspace for the send queue | The dbspace must already exist. The path for the send queue can be no longer than 128 bytes. The *send_space* cannot be a temporary dbspace. | Identifiers, p. 8-8 |
| *timeout* | Idle timeout for this replication server | Must be an integer number of minutes. 0 indicates no timeout. The maximum value is 32 kilobytes; 0 indicates no timeout. | Integer |

The options for **cdr define server** have the meanings shown in the following table.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --ats= | -A | Activates aborted-transaction spooling for replicate transactions that fail to be applied to the target database. For more information, see Chapter 10. |
| --idle= | -i | Causes an inactive connection to be terminated after *timeout* minutes. If timeout is 0, the connection does not timeout. The default value is 0. |
| --recv= | -r | Assigns the receive queue to the dbspace *recv_space*. If this option is omitted, the receive queue is in the root dbspace. |
| --ris= | -R | Activates row-information spooling for replicate row data that fails conflict resolution or encounters replication-order problems. For more information, see Chapter 10. |
| --send= | -s | Assigns the send queue to the dbspace *send_space*. If this option is omitted, the receive queue is in the root dbspace. |

## Examples

The following example connects to the database server **stan**, initializes Enterprise Replication, and sets the idle time-out of 500 minutes. The example also specifies that the send queue should be stored in **dbspace1** and the receive queues should be stored in **dbspace2**. Any files that ATS generates will go into directory **/cdr/ats**.

```
cdr define server --connect=stan --idle=500 --send=dbspace1 \
    --recv= dbspace2 --ats /cdr/ats --init g_stan
```

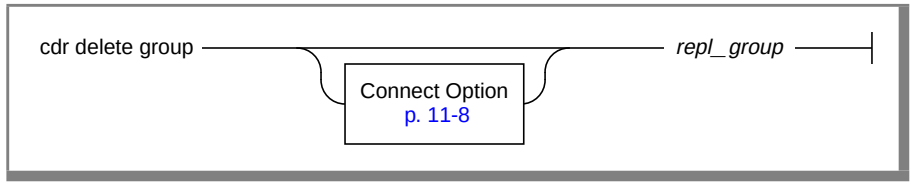
The following example connects to the database server **oliver**, initializes Enterprise Replication, synchronizes its catalogs with the catalogs on database server **stan**, and defines the database server **oliver** with an idle time-out of 600 minutes. The send queue will be created in **dbspaceA** and the receive queues will be created in **dbspaceB**. Also, any files that ATS generates will go into directory **/cdr/ats**.

```
cdr define server -c oliver-i 600 -s dbspaceA -r dbspaceB \
     -A /cdr/ats -I -S stan oliver
```

# cdr delete group

The **cdr delete group** command deletes a replicate group.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_group* | Name of replicate group to delete | The group must exist. | Identifiers, p. 8-8 |

## Usage

The **cdr delete group** command deletes the replicate group *repl_group* from the global catalog.

The **cdr delete group** command does not affect the replicates or associated data. When a replicate group is deleted, the individual replicates within the group are unchanged.

## Example

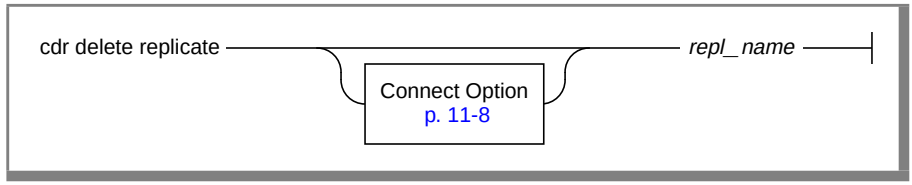
The following example connects to the default database server and deletes the replicate group **accounts**:

```
cdr delete group accounts
```

# cdr delete replicate

The **cdr delete replicate** command deletes a replicate from the global catalog.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_name* | Name of the replicate to delete | The replicate must exist. | Identifiers, p. 8-8 |

## Usage

The **cdr delete replicate** command deletes the replicate *repl_name* from the global catalog. All replication data for the replicate is purged from the send queue at all participating database servers. If *repl_name* is contained in a *replicate group*, this command fails. You must remove the replicate from the replicate group before you can delete it.

To add or delete a participant using ERM, refer to "Deleting a Replicate" on page 8-13.

## Example

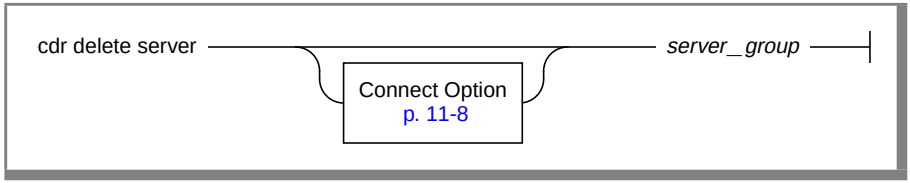
The following command connects to the default database server (**$INFORMIXSERVER**) and deletes the replicate **smile**:

```
cdr del rep smile
```

# cdr delete server

The **cdr delete server** command deletes a database server from the global catalog.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *server_group* | Name of server group to remove from the global catalog. | The server group must be currently defined in Enterprise Replication. | Database Server Group, p. 11-12 |

## Usage

The **cdr delete server** command deletes the database server in *server_group* from the global catalog, removes the database server from all participating replicates, and purges all replication data from the send queues for the specified database server. The command shuts down Enterprise Replication on the database server and removes the global catalog from the database server.

When you delete an Enterprise Replication server, you must issue the **cdr delete serve**r command *twice*. The first **cdr delete server** removes the Enterprise Replication server from the local global catalog and removes the Enterprise Replication connection to other hosts. The second **cdr delete server** removes the Enterprise Replication server from the replication server that you want to delete.

You can issue the **delete server** command from any replication server. The only limitation is that one cannot delete a server with children. You must delete the children of a server before deleting the parent server.

## Examples

This example removes the server **g_italy** from the replication environment. Assume that you issue the commands from the replication server **g_usa**.

```
cdr delete server g_italy
cdr delete server -c italy g_italy
```

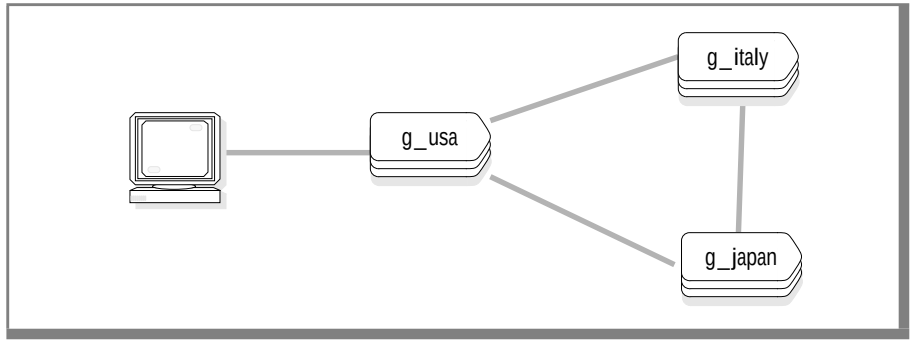
The first command, **cdr delete server g_italy**, performs the following actions:

- Removes **g_italy** from the **usa** global catalog
- Drops the connection from **g_usa** to **g_italy**
- Removes **g_italy** from all participating replicates
- Purges the replication data destined for **g_italy** from send queues.
- Broadcasts this delete server command to all other servers (other than **g_italy**) so that they can perform the same actions

The second command, **cdr delete server -c italy g_italy**, connects to server **italy**, and removes Enterprise Replication from **italy**. That is, it removes the **syscdr** database and removes or stops other components of Enterprise Replication.

Figure 11-1 shows a replication environment with three replication servers, **g_usa**, **g_italy**, and **g_japan**.

***Figure 11-1***
*Three Replication Servers*



To remove Enterprise Replication from this environment, issue the following commands from the computer where the **usa** replication server resides:

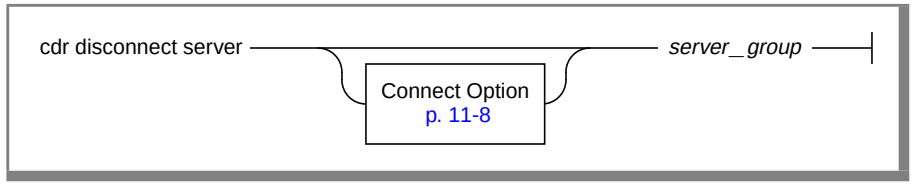
```
1  cdr delete server g_italy
2  cdr delete server -c italy g_italy
3  cdr delete server g_japan
4  cdr delete server -c g_japan
5  cdr delete server g_usa
```

Line 1 removes connections between the **italy** replication server and all other servers in the replication system (**usa** and **japan**) and removes any queued data. Line 2 removes all replication information (including the **syscdr** database) from the **italy** database server. Similarly, lines 3 and 4 remove the **japan** replication server. Finally, line 5 removes the replication information from the **usa** replication server itself.

# cdr disconnect server

The **cdr disconnect server** command stops a server connection.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *server_group* | Name of the server group to disconnect | The server group must be currently active in Enterprise Replication. | Database Server Group, p. 11-12 |

## Usage

The **cdr disconnect server** command drops the connection between *server_group* list and the server specified in the **Connect** option. If the **Connect** option is omitted, the command drops the connection between *server_group* and the default database server (**$INFORMIXSERVER**).

## Example

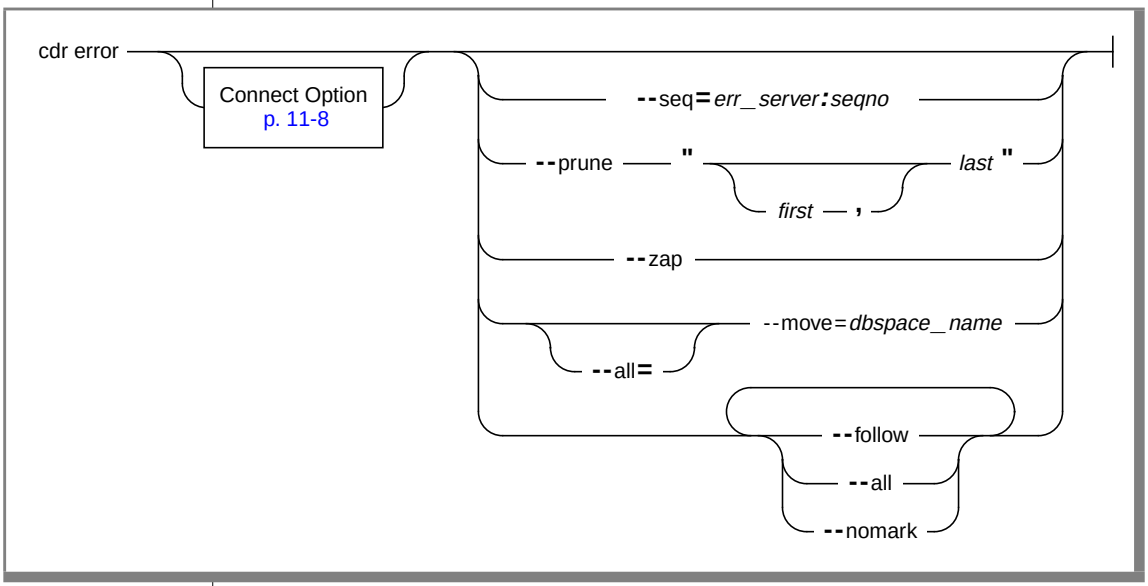
The following example drops the connection between the default database server (**$INFORMIXSERVER**) and the server group **g_store1**:

```
cdr disconnect server g_store1
```

# cdr error

The **cdr error** command manages the error table and provides convenient displays of errors.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *dbspace_name* | The dbspace where the error table is stored. | The dbspace must exist. | Identifiers, p. 8-8 |
| *err_server* | Name of database server group that holds the error table | The server must be registered for Enterprise Replication. | Identifiers, p. 8-8 |
| *first* | Start date for a range | Must be a valid date and time. | Time of Day, p. 11-15 |
| *last* | Ending date for range | Must be a later date and time than first. | Time of Day, p. 11-15 |
| *seqno* | Sequence number of a specific error | Must be the number of an error in the error table. | Integer |

The options for **cdr error** have the meanings shown in the following table.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| (no options specified) | | Print the current list of errors. After the errors have been printed, mark them as *reviewed*. |
| --all | -a | When used with the -move option, move the error table to *dbspace* on all defined servers. Otherwise, print all errors, including those already reviewed. |
| --follow | -f | Continuously monitor the error table. |
| --move | -m | Move the error table to the dbspace *dbspace_name*. |
| --nomark | -n | Do not mark errors as *reviewed*. |
| --prune | -p | Prune the error table to those times in the range from *first* to *last*. If *first* is omitted, then all errors earlier than *last* are removed. |
| --seq | -s | Remove the (single) error specified by *server:seqno* from the error table. |
| --zap | -z | Remove all errors from the error table. |

## Usage

The **cdr error** command allows you to examine replication errors on any replication server. Sometimes a CLU command can succeed on the server on which it is executed but fail on one of the remote servers. For example, if you execute **cdr define replicate** on **server1** but the table name is misspelled on **server2**, the command succeeds on **server1** and appears to have completed successfully. You can use **cdr error -c server2** to see why replication is not occurring.

The **cdr error** command also allows you to administer the **cdr error** table remotely. The reviewed flag lets you watch for new errors while keeping the old errors in the table. For example, you could run **cdr error** periodically and append the output to a file.

## Examples

The following command displays the current list of errors on database server **hill**. After the errors are displayed, Enterprise Replication marks the errors as *reviewed*.

```
cdr error --connect=hill
```

The following command connects to the database server **lake** and removes from the error table all errors that occurred before the time when the command was issued:

```
cdr error -c lake --zap
```

The following command deletes all errors from the error table that occurred at or before 2:56 in the afternoon on May 1, 1998:

```
cdr error -p "1998-05-01 14:56:00"
```

The following command deletes all errors from the error table that occurred at or after noon on May 1, 1998 and before or at 2:56 in the afternoon on May 1, 1998:

```
cdr error -p "1998-05-01 14:56:00,1998-05-01 12:00:00"
```

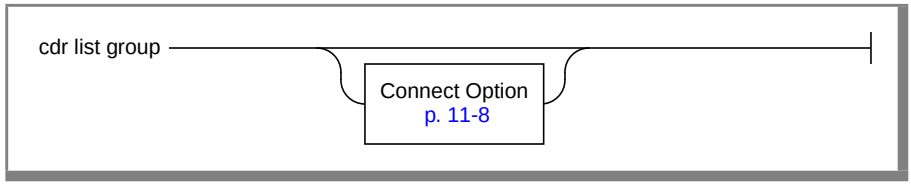
The following command moves the error table to **dbspace1** on all defined servers:

```
cdr error -a -m dbspace1
```

# cdr list group

The **cdr list group** command displays information about the groups on the current server.

## Syntax




## Usage

The **cdr list group** command displays information about replication groups.

You do not need to be user **informix** to use this command.

## Example

The following example displays a list of the groups on the current server:

```
cdr list group
```

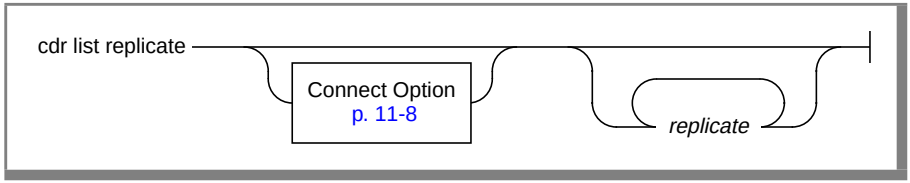
Sample output:

```
GROUP       SEQ  PARTICIPANTS
------------------------------------------------------
weekly       Y   rep1, rep2, rep3
NewGroup     N   one, three, seven
```

# cdr list replicate

The **cdr list replicate** command displays information about the replicates on the current server.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *replicate* | Name of the replicate(s) | The replicate(s) must exist. | Identifiers, p. 8-8 |

## Usage

The **cdr list replicate** command displays information about replicates. If no replicates are named, the command lists all replicates on the current server. If one or more replicates are named, the command displays detailed information about those replicates.

You do not need to be user **informix** to use this command.

## Example

The following example displays a list of the replicates on the current server:

```
cdr list replicate
```

The output from the previous command might be the following:

```
REPLICATE  STATE      CONFLICT      FREQUENCY  OPTIONS
------------------------------------------------------------------------------
myrep      INACTI     ignore        immediate  transaction,ris,ats
rOne       ACTIVE     ignore        immediate  transaction,ris,ats
rTwo       ACTIVE     ignore        immediate  transaction,ris,ats
```

The **STATE** column can include the following values:

- ACTIVE (replicate active)
- DEFN FAI (definition failure)
- INACT (inactive)
- PEND (pending delete)
- QUIES (quiescent)
- SUSP (suspend)

The following example specifies the names of replicates:

```
cdr list repl rOne rTwo
```

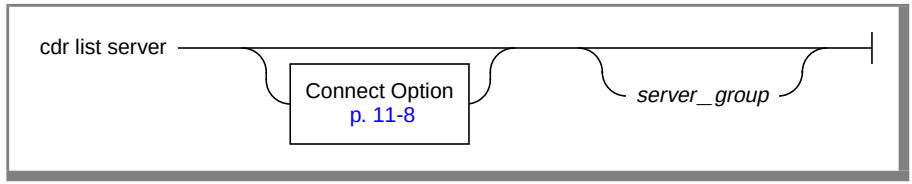
The following output might result from the previous command:

```
REPLICATE  SERVER   STATE   TABLE      SELECT
--------------------------------------------------------------------------------
rOne       g_srv1   INACT  customer    Select * from informix.state
rOne       g_srv2   INACT  customer    Select * from informix.state
rTwo       g_srv1   INACT  customer    Select address1, address2, city, company,
customer_num, fname, lname, phone, state, zipcode from informix.customer
rTwo g_srv2 INACT customer Select address1, address2, city, company,
customer_num, fname, lname, phone, state, zipcode from informix.customer
```

# cdr list server

The **cdr list server** command displays a list of the Enterprise Replication servers that are visible to the current server.

## Syntax




## Usage

The **cdr list server** command displays information about servers. You do not need to be user **informix** to use this command.

In ERM, this information is displayed from **Server→Server Properties**.

### *Listing All Enterprise Replication Servers*

When no server-group name is given, the **cdr list server** command lists all database server groups that are visible to the current replication server.

For example, `cdr list server` might give the following output:

```
SERVER    ID    STATE    STATUS      CONNECTION CHANGED
------------------------------------------------------------
g_svr1    33    Active    Connected  Mar 19 17:31:51 1999
g_svr2    34    Active    Dropped    Mar 19 18:04:28 1999
g_svr3    22    Active
```

#### *The SERVER and ID Columns*

The SERVER and ID columns display the name of an Enterprise Replication server and its replication ID.

### The STATE Column

The STATE column can have the following values:

- Active
- Quiescent
- Suspended

### The STATUS Column

The STATUS column can have the following values:

- Connected
- Disconnect
- Dropped
- Error Detected
- Local
- Timeout

### The CONNECTION Column

The CONNECTION column displays the most recent time that the status of the server connection was changed.

## Showing Details about a Single Replication Server

When the **cdr list server** command includes the name of a database server group, the command displays the attributes of that database server. For example, `cdr list server g_usa` might give the following output:

```
NAME      ID     ATTRIBUTES
--------------------------------------
g_usa      3   timeout=15 sendq=rootdbs recvq=rootdbs
atsdir=/tmp risdir=/tmp
```

## Examples

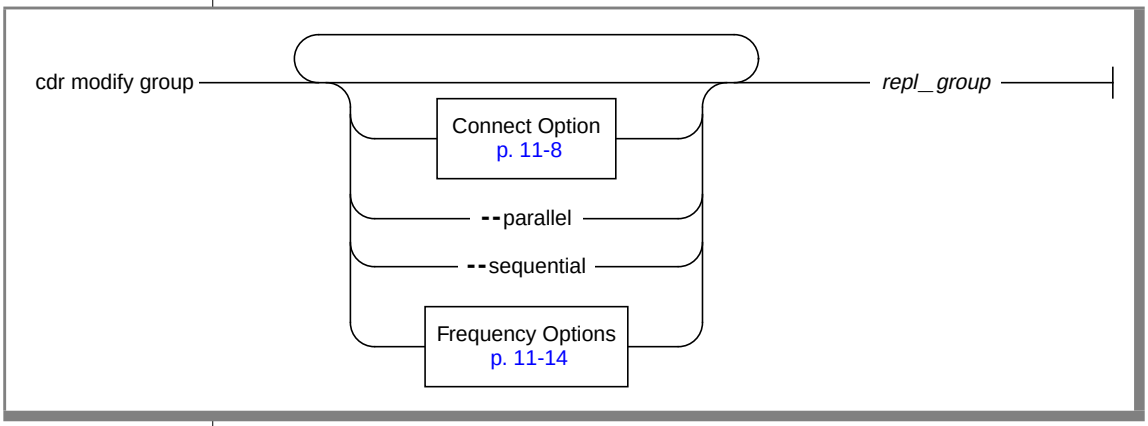
In the following examples, **usa** and **italy** are root servers, **denver** and **boston** are nonroot servers, and **miami** is a leaf server. The **usa** server is the parent of **denver** and **boston**, and **boston** is the parent of **miami**.

```
NAME     ID     ATTRIBUTES
----------------------------------------
cdr list server g_usa
g_usa     1 timeout=15 hub sendq=rootdbs recvq=rootdbs
atsdir=/tmp risdir=/tmp

cdr list server -c denver g_denver
g_denver  27 root=g_usa sendq= recvq= atsdir=/ix/myats
risdir=

cdr list server -c italy g_denver
g_denver  27 root=g_usa forward=g_usa sendq= recvq=
atsdir=/ix/myats risdir=

cdr list server g_miami
g_miami   4 root=g_boston leaf sendq= recvq= atsdir= risdir=
```

# cdr modify group

The **cdr modify group** command modifies a replicate group.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *rep_group* | Name of replicate group to modify | The group must exist. | Identifiers, p. 8-8 |

## Usage

The **cdr modify group** command modifies the attributes of the replicate group *rep_group*. Attributes of the group that are not specifically mentioned remain unchanged. To add or delete replicates to a group, use the **cdr change group** command on page 11-20.

The options in **cdr modify group** have the meanings shown in the following table.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --parallel | -p | Process members of the group in parallel. |
| --sequential | -s | Process members of the group sequentially. |

To change between parallel and sequential processing from ERM, refer to "Viewing Replicate Group Properties" on page 8-24.

## Example

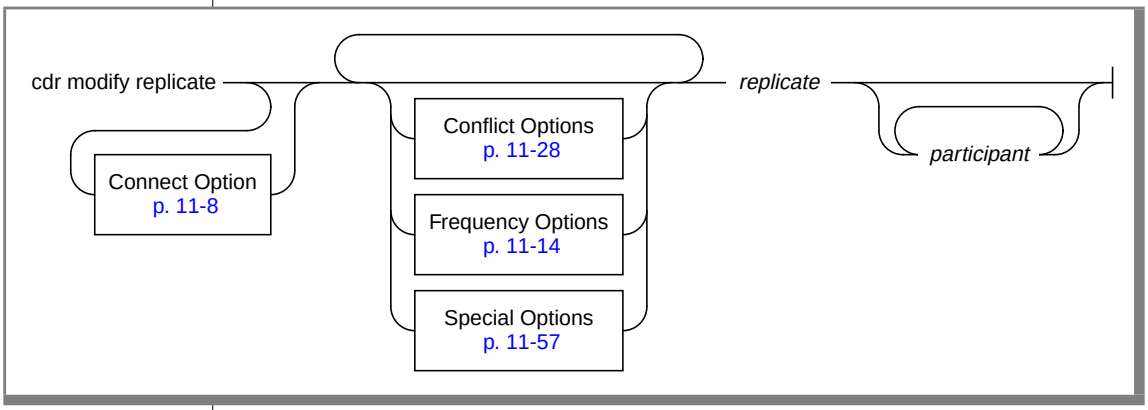
The following example connects to the default server (**$INFORMIXSERVER**) and modifies the replicate group **accounts** to process replication data sequentially for replicates in **accounts**:

```
cdr modify group --sequential accounts
```

# cdr modify replicate

The **cdr modify replicate** command modifies replicate attributes.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *participant* | Name of a participant in the replication | The participant must be a member of the replicate. | Participant Name, p. 11-9 |
| *replicate* | Name of the replicate to modify | The replicate name must exist. | Identifiers, p. 8-8 |

## Usage

The **cdr modify replicate** command modifies a replicate. You can change the attributes of a replicate or of one or more participants in the replicate. You can also change the mode of a participant. If the command does not specify participants, the changes apply to all participants in the replicate.

For attribute information, see "cdr define replicate" on page 11-27.

To add or delete a participant, see "cdr change replicate" on page 11-22.

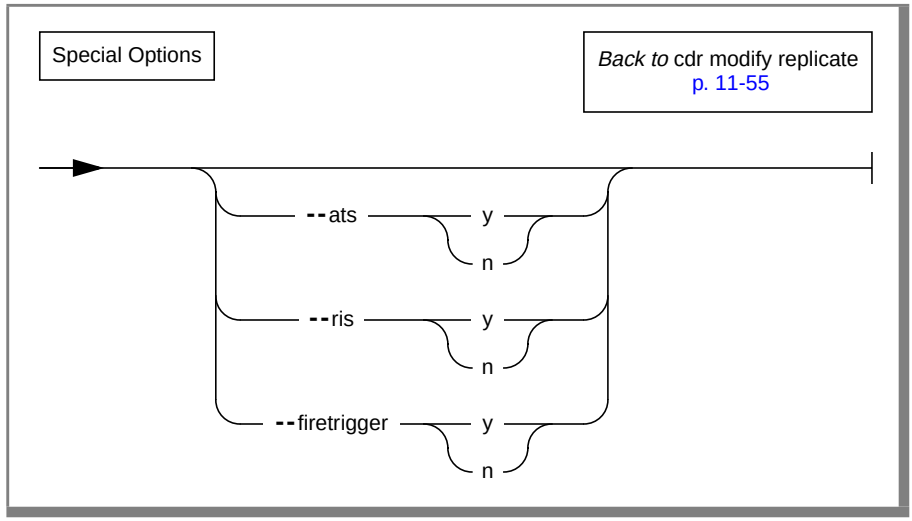
If you change the conflict-resolution rule with **cdr modify replicate**, you must also explicitly state SCOPE parameter, even if the SCOPE value does not change.

### Restrictions

The attributes for **cdr modify replicate** are the same as the attributes for **cdr define replicate**, with the following exceptions:

- You cannot change the canonical float format.
- You cannot change the conflict resolution from ignore to a non-ignore option or from non-ignore to ignore. However, you can change from time-stamp resolution to procedure resolution or from procedure resolution to time stamp.
- The ATS, RIS, and Fire Triggers options require a yes (y) or no (n) argument.

## Special Options




The special options for **cdr define replicate** have the meanings shown in the following table.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --ats | -A | Activate (y) or deactivate (n) aborted-transaction spooling for replicate transactions that fail to be applied to the target database. For more information, see Chapter 10. |
| --firetrigger | -T | Cause the rows inserted by this replicate to fire (y) or not fire (n) triggers at the destination. |
| --ris | -R | Activate (y) or deactivate (n) row-information spooling for replicate row data that fails conflict resolution or encounters replication-order problems. For more information, see Chapter 10. |

## Examples

The following example modifies the frequency attributes of replicate **smile** to replicate every 5 hours:

```
cdr modify repl -every=300 smile
```

The following example modifies the frequency attributes of replicate **smile** to replicate daily at 1:00 A.M.:

```
cdr modify repl -a 01:00 smile
```

The following example modifies the frequency attributes of replicate **smile** to replicate on the last day of every month at 5:00 A.M., to generate ATS files and not to fire triggers:

```
cdr modify repl -a L.5:00 -A y -T n smile
```

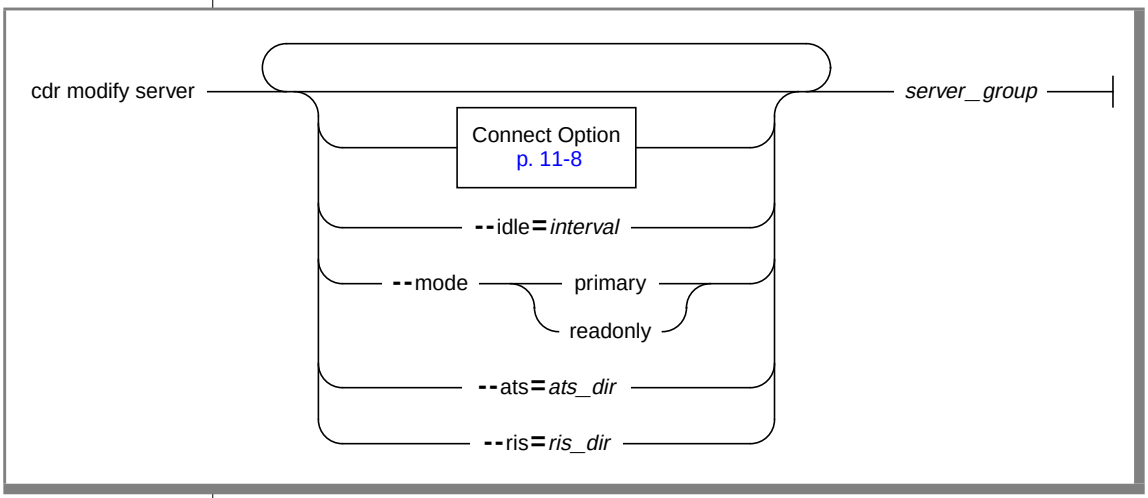
The following example changes the mode of the first participant listed to read-only and the mode of the second to primary. If **server1** needed to be taken for some reason, you might want to make this type of change.

```
cdr mod repl smile "R db1@server1:antonio.table1" \
                   "P db2@server2:carlo.table2"
```

# cdr modify server

The **cdr modify server** command modifies the Enterprise Replication attributes of a database server.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *ats_dir* | Name of aborted-transaction spooling directory | Must be a full pathname. | Directory name on your operating system. |
| *interval* | Idle time-out for this server | Must be an integer >=0. 0 = no timeout. | Integer number of minutes. |
| *ris_dir* | Name of the row-information spooling directory | Must be a full pathname. | Directory name on your operating system. |
| *server_group* | Name of a server group to modify | The server group must be defined in Enterprise Replication. | Database server Group, p. 11-12 |

## Usage

The **cdr modify server** command modifies the replication server *server_group*.

The options for **cdr modify server** have the meanings that the following table shows.

| Options | | |
|---|---|---|
| **Long Form** | **Short Form** | **Meaning** |
| --ats | -A | Activate aborted-transaction spooling for replicate transactions that fail to be applied to the target database. For more information, see Chapter 10. |
| --idle | -i | Causes an inactive connection to be terminated after *idle* time. |
| --mode | -m | Change the mode of all replicates using this server to primary or to read-only. You can use the abbreviations `p` and `r`, for primary and read only, respectively. |
| --ris | -R | Activates row-information spooling for replicate-row data that fails conflict resolution or encounters replication-order problems. For more information, see Chapter 10. |

## Examples

The following example connects to the database server **paris** and modifies the idle time-out of server group **g_rome** to 10 minutes. ATS files go into the directory **/cdr/atsdir**.

```
cdr modify server-c paris -i 10 -A /cdr/atsdir g_rome
```

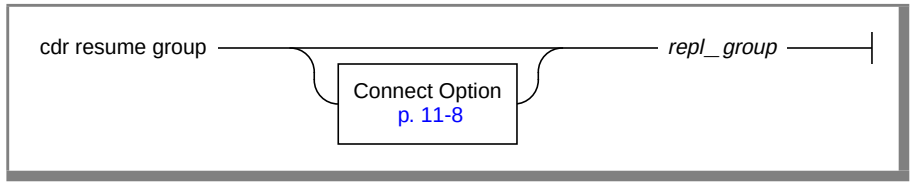
The following example connects to the default database server and sets the modes of all participants on **g_geometrix** to primary:

```
cdr modify server -m primary g_geometrix
```

# cdr resume group

The **cdr resume group** command resumes delivery of replication data for replicates defined in a replicate group.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_group* | Name of replicate group to resume | The group must be suspended. | Identifiers, p. 8-8 |

## Usage

The **cdr resume group** command causes all replicates contained in the replicate group *repl_group* to enter the active state for all participants.

For more information about replication states, refer to "Changing Replicate States" on page 8-14.

## Example

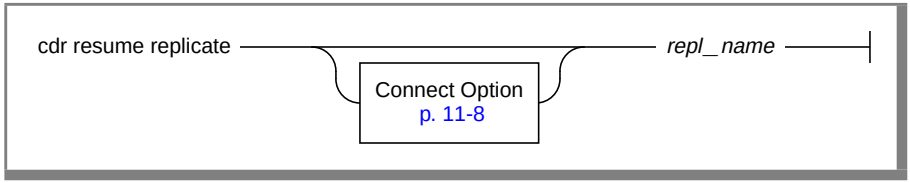
The following example connects to the default database server (**$INFORMIXSERVER**) and resumes the replicate group **accounts**:

```
cdr resume group accounts
```

# cdr resume replicate

The **cdr resume replicate** command resumes delivery of replication data.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_name* | Name of the replicate to change to active state. | The replicate must not be a member of a replicate group and must be suspended. | Identifiers, p. 8-7 |

## Usage

The **cdr resume replicate** command causes all participants in the replicate *repl_name* to enter the active state.

If the replicate *repl_name* is a member of a replicate group, this command fails. When *repl_name* is in a replicate group, use the **cdr resume group** command on page 11-61.

For more information on replicate states, refer to "Changing Replicate States" on page 8-14.

## Example

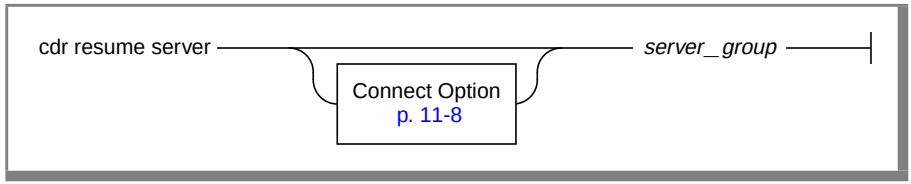
The following example connects to the default database server (**$INFORMIXSERVER**) and resumes the replicate **smile**:

```
cdr resume repl smile
```

# cdr resume server

The **cdr resume server** command resumes delivery of replication data to a database server.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *server_group* | Name of server group to resume | The server group must be defined for replication and must be suspended. | Server Group, p. 11-12 |

## Usage

The **cdr resume server** command restarts suspended replication to the server *server_group*. The server must have been previously suspended with the **cdr suspend server** command on page 11-77.

## Example

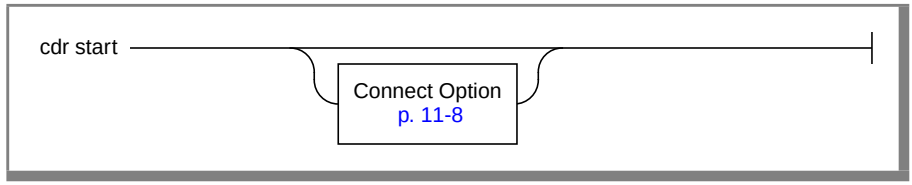
The following example resumes replication on the server groups **g_utah** and **g_iowa**:

```
cdr resume serv g_utah g_iowa
```

# cdr start

The **cdr start** command starts Enterprise Replication processing.

## Syntax




## Usage

Use **cdr start** to restart Enterprise Replication after you stop it with **cdr stop**. When you issue **cdr start**, Enterprise Replication brings up all connections to other adjacent replication servers and starts sending previously queued data. Replication servers, replicates, and groups that were suspended before the **cdr stop** command was issued remain suspended; no data is sent for the suspended servers, replicates, or groups.

Enterprise Replication resumes evaluation of the logical log (if required for the instance of Enterprise Replication) at the *replay* position. The replay position is the position where Enterprise Replication stops evaluating the logical log when **cdr stop** is executed. If the evaluation process is running and the logical log ID for the replay position no longer exists when Enterprise Replication is started, then the restart partially fails (the database server log contains an error message stating that the replay position is invalid). If the restart partially fails, no database updates performed on the local database server are replicated.

***Warning:*** *Informix recommends that you issue **cdr start** and **cdr stop** with extreme caution. Use these commands during a period of little or no database activity and keep the downtime to a minimum. Transactions that occur during this period are not replicated and hence, tables will be unsynchronized.*

## Example

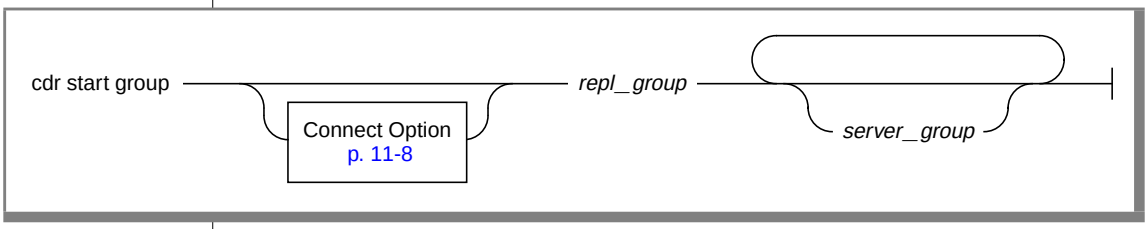
The following example restarts Enterprise Replication processing on database server **utah**:

```
cdr start -c utah
```

# cdr start group

The **cdr start group** command starts the capture and transmittal of replication transactions for the replicates contained in a replicate group.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_group* | Name of replicate group to start | The group must exist. | Identifiers, p. 8-8 |
| *server_group* | Name(s) of server group(s) on which to start the group | The server(s) must be defined for Enterprise Replication. | Database Server Group, p. 11-12 |

## Usage

The replicates defined in the replicate group *repl_group* enter the active state (capture-send) on the database servers in *server_group*. If no server group is specified, all database servers in the replication system enter active state.

## Example

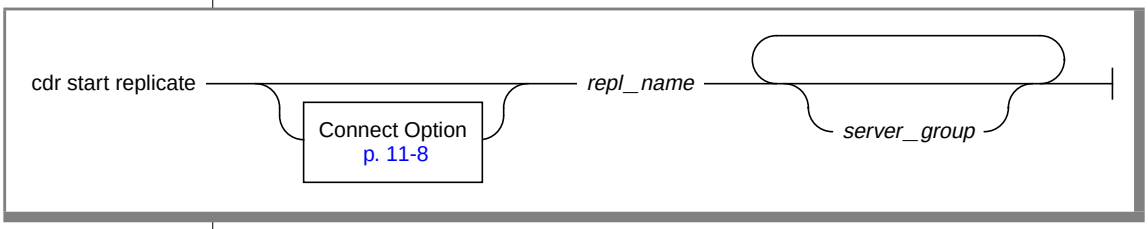
Connect to the default database server and start the replicate group **accounts** on the server groups **g_hill** and **g_lake**.

```
cdr start group accounts g_hill g_lake
```

# cdr start replicate

The **cdr start replicate** command starts the capture and transmittal of replication transactions.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_name* | Name of the replicate to start | The replicate must exist. | Identifiers, p. 8-8 |
| *server_group* | Name of server group(s) on which to start the replicate | The server group(s) must be defined for Enterprise Replication. | Database Server Group, p. 11-12 |

## Usage

The **cdr start replicate** command causes the replicate *repl_name* to start on the server(s) *target* server. If no *target* server is specified, the *repl_name* starts on all servers that are included in the replicate. When a replicate *starts* on a server, the participant on that server enters the active state (data is captured and sent). Thus, a replicate can have both active and inactive participants. When at least one participant is active, the replicate is active.

If *repl_name* is a member of a replicate group, this command fails. Instead, use the **cdr start group** command on page 11-66.

## Example

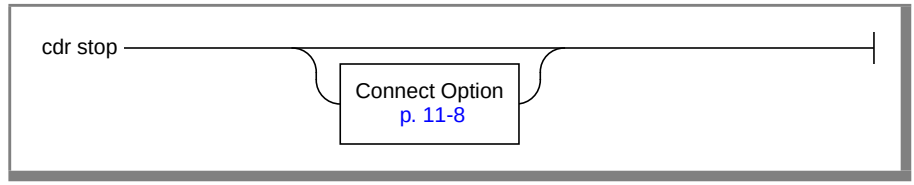
The following command starts the replicate **accounts** on the server groups **g_svr1** and **g_svr2**:

```
cdr start replicate accounts g_svr1 g_svr2
```

# cdr stop

The **cdr stop** command stops Enterprise Replication processing.

## Syntax




## Usage

In most situations, Enterprise Replication starts when **cdr define server** is first executed. The replication threads remain running until the database server is shut down, or until the local database server is deleted with the **cdr delete server** command. If you shut down the database server while Enterprise Replication is running, replication begins again when you restart the database server.

Under rare conditions, users might want to temporarily stop the Enterprise Replication processing without stopping the database server. The **cdr stop** command shuts down all Enterprise Replication threads in an orderly manner. When the shutdown of Enterprise Replication is complete, the message `CDR shutdown complete` appears in the database server log file.

After issuing the **cdr stop** command, replication threads remain stopped (even if the database server is stopped and restarted) until a **cdr start** command is issued.

***Warning:*** *If you issue **cdr stop** and database activity continues, the database server from which the command is issued and the other database servers participating in replicates will become inconsistent. To ensure consistency, verify that no database update activity occurs while Enterprise Replication is stopped.*

## Example

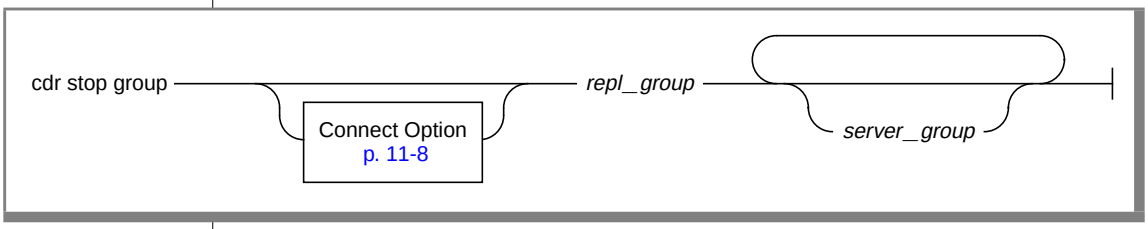
The following example stops Enterprise Replication processing on database server **paris**. Processing does not resume until a **cdr start** command restarts it.

```
cdr stop -c paris
```

# cdr stop group

The **cdr stop group** command stops capture and transmittal transactions for replicates contained in a replicate group.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_group* | Name of replicate group to stop | The group must be a currently active group. | Identifiers, p. 8-8 |
| *server_group* | Name of server group on which to stop the replicate group | The server group(s) must be defined for Enterprise Replication. | Database Server Group, p. 11-12 |

## Usage

The **cdr stop group** command causes all replicates in the replicate group *repl_group* to enter the *inactive* state *(*no capture, no send*)* on the database servers in the *server_group* list. If the *server_group* list is omitted, the replicate group *repl_group* enters the inactive state for all database servers participating in the replicate group.

## Example

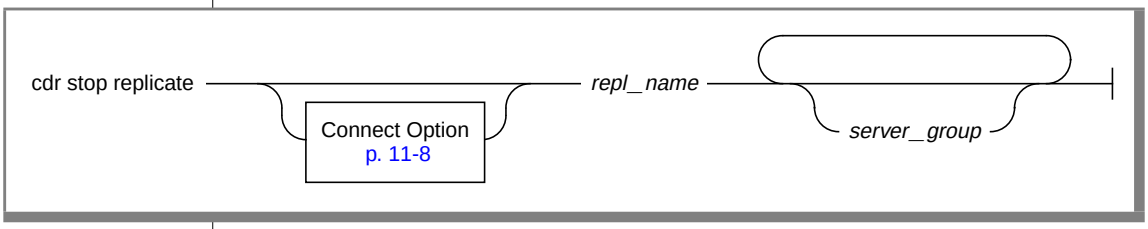
The following example connects to the database server **paris** and stops the group **accounts** on server groups **g_utah** and **g_iowa**:

```
cdr stop group --connect=paris accounts g_utah g_iowa
```

# cdr stop replicate

The **cdr stop replicate** command stops the capture and transmittal of transactions for replication.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_name* | Name of the new replicate | The replicate must be active and not in a replicate group. | Identifiers, p. 8-8 |
| *server_group* | List of server groups on which to stop the replicate | The server group(s) must be defined for Enterprise Replication. | Database Server Group, p. 11-12 |

## Usage

The **cdr stop replicate** command causes the replicate *repl_name* to enter the inactive state (no capture, no send) on the replicate servers in the *server_group* list. That is, the participants that are on the specified replicate servers are no longer active.

If the *server_group* list is omitted, the replicate enters the inactive state on all database servers participating in the replicate.

If the replicate *repl_name* is contained in a replicate group, this command fails. When *repl_name* is in a replicate group, use the **cdr stop group** command on page 11-70.

## Example

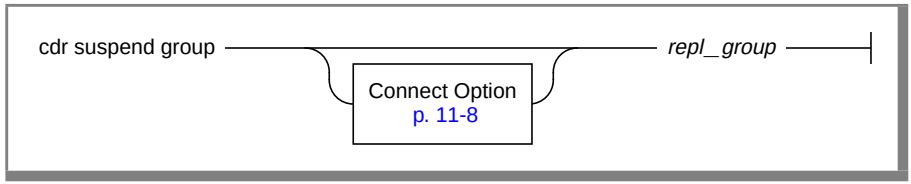
The following command connects to the database server **lake** and stops the replicate **aRepl** on server groups **g_server1** and **g_server2**:

```
cdr stop replicate -c lake aRepl g_server1 g_server2
```

# cdr suspend group

The **cdr suspend group** command suspends delivery of replication data for replicates contained in a replicate group.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_group* | Name of replicate group to suspend | The group must be a currently active group. | Identifiers, p. 8-8 |

## Usage

The **cdr suspend group** command causes the replicates in the replicate group *repl_group* to enter the suspend state. Information is captured, but no data is sent for any replicate in the group. The data is queued to be sent when the group is resumed.

You cannot remove replicates from a replicate group when the group is suspended.

***Warning:*** *When a replicate group is suspended, Enterprise Replication holds the replication data in the send queue until the group is resumed. If lots of data is generated for the group while it is suspended, the send queue space can fill, causing data to be lost.*

***Warning:*** *Enterprise Replication does not synchronize transactions if a replicate group is suspended. For example, a transaction that updates tables X and Y will be split if replication for table X is suspended.*

## Example

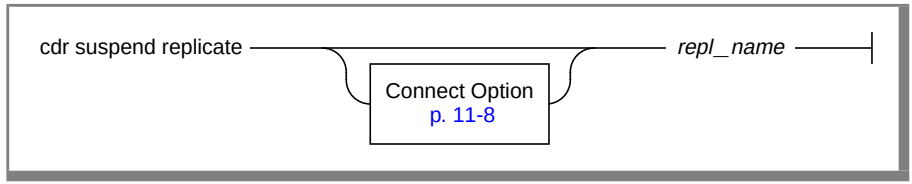
The following example connects to the default database server (**$INFORMIXSERVER**) and suspends the replicate group **accounts**:

```
cdr suspend group accounts
```

# cdr suspend replicate

The **cdr suspend replicate** command suspends delivery of replication data.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *repl_name* | Name of the new replicate | The replicate must be active. | Identifiers, p. 8-8 |

## Usage

The **cdr suspend replicate** command causes the replicate *repl_name to* enter the suspend state (capture, no send) for all participants.

If the replicate *repl_name* is contained in a replicate group, this command fails. When *repl_name* is in a replicate group, use the **cdr suspend group** command on page 11-73.

You cannot add a replicate that is suspended to a replicate group even if the replicate group is also suspended.

***Warning:*** *When a replicate is suspended, Enterprise Replication holds the replication data in the send queue until the replicate is resumed. If lots of data is generated for this replicate while it is suspended, the send queue space can fill, causing data to be lost.*

***Warning:*** *Enterprise Replication does not synchronize transactions if a replicate is suspended. For example, a transaction that updates tables X and Y will be split if replication for table X is suspended.*

## Example

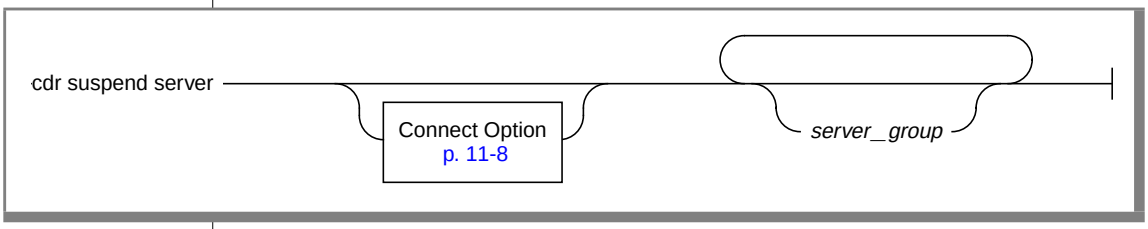
The following example connects to the database server **stan** and suspends the replicate **house**:

```
cdr sus repl --connect stan house
```

# cdr suspend server

The **cdr suspend server** command suspends the delivery of replication data to a database server.

## Syntax




| Element | Purpose | Restrictions | Syntax |
|---|---|---|---|
| *server_group* | Name of server group to suspend | The server group must be currently active in Enterprise Replication. | Database Server Group, p. 11-12 |

## Usage

The **cdr suspend server** command suspends delivery of replication data from the current database server specified to the database servers included in the *server_group* list. The *current database server* is server specified in the **Connect** option or the default server (**$INFORMIXSERVER**).

The connection to the suspended server is unaffected, and control and acknowledge messages continue to be sent to that server. Replication data continues to all other database servers.

If the *server_group* list is omitted, the command suspends replication of data to all database servers participating in the Enterprise Replication system.

## Example

The following example connects to the default server (**$INFORMIXSERVER**) and suspends the servers **g_iowa**, **g_ohio**, and **g_utah**:

```
cdr suspend serv g_iowa, g_ohio, g_utah
```

Chapter

# 12

# API Functions


In This Chapter . . . . . 12-3

Basic Information for APIs . . . . . 12-3
- Database Server Groups . . . . . 12-3
- Time Specifications . . . . . 12-5
- Identifiers . . . . . 12-5
- API Structures . . . . . 12-5
- Severe-Error Table . . . . . 12-7
  - Information Maintained in the Severe-Error Table . . . . . 12-7
  - Retrieving Information from the Severe-Error Table . . . . . 12-8

Enterprise Replication APIs . . . . . 12-8
- cdr_connect() . . . . . 12-10
- cdr_define_group() . . . . . 12-11
- cdr_define_repl() . . . . . 12-14
- cdr_define_serv() . . . . . 12-18
- cdr_delete_group() . . . . . 12-20
- cdr_delete_repl() . . . . . 12-21
- cdr_delete_serv() . . . . . 12-22
- cdr_error_reviewed() . . . . . 12-23
- cdr_modify_group() . . . . . 12-24
- cdr_modify_repl() . . . . . 12-25
- cdr_modify_replmode() . . . . . 12-27
- cdr_modify_serv() . . . . . 12-28
- cdr_modify_servmode() . . . . . 12-29
- cdr_move_errortab() . . . . . 12-30
- cdr_participate_group() . . . . . 12-31
- cdr_participate_repl() . . . . . 12-32
- cdr_prune_errors() . . . . . 12-33

cdr_prune_single_error() . . . . . 12-34
cdr_resume_group() . . . . . 12-35
cdr_resume_repl() . . . . . 12-36
cdr_resume_serv() . . . . . 12-37
cdr_start_cdr() . . . . . 12-38
cdr_start_group() . . . . . 12-39
cdr_start_repl() . . . . . 12-40
cdr_stop_cdr() . . . . . 12-41
cdr_stop_group() . . . . . 12-42
cdr_stop_repl() . . . . . 12-43
cdr_suspend_group() . . . . . 12-44
cdr_suspend_repl() . . . . . 12-45
cdr_suspend_serv() . . . . . 12-46
Example . . . . . 12-47

## In This Chapter

The application programming interface (API) allows Enterprise Replication to provide a single control point to configure and control replication objects. Replication objects consist of database servers, replicates, and replicate groups.

# Basic Information for APIs

The next sections contain information that applies to all the API functions that this chapter discusses.

## Database Server Groups

All API functions except the **cdr_connect()** option use the *database server group* name instead of the more familiar *database server* name for all references to database servers. Informix database servers allow you to create a database server group that includes several database server aliases. Database server groups are useful, for example, if multiple connection protocols are available for connecting to a single database server. In practice, a database server group usually contains only one database server name.

If you use both DBSERVERNAME and DBSERVERALIASES, DBSERVERNAME should refer to the network connection and not to a shared-memory connection. For information about database server aliases, refer to the *Administrator's Guide for Informix Dynamic Server 2000*.

UNIX

On UNIX, the **sqlhosts** file defines database server groups. The first line in the following example specifies a database server group. The second line specifies a database server that belongs to the group.

```
g_svrA     group        -          -          i=92
svrA       ontlitcp     smoke      svcname    g=g_srvA
```

♦

WIN NT

With Dynamic Server, Version 7.31, use the IECC Add Database Servers wizard to prepare the SQLHOSTS connectivity information.

With Dynamic Server, Version 9.2, refer to the documentation notes described in "Documentation Notes, Release Notes, Machine Notes" on page 15 of the Introduction for information about preparing the SQLHOSTS connectivity information.♦

This manual uses the convention that the name of a database server group is **g_** followed by the name of a database server that is in the group. This use of **g_** is only a convention; **g_** is not required syntax.

If you prefer, you can give your database server group the same name as the first database server in the group. In that case, the term *server* can refer to either the database server group or the database server. In the following example, the first line defines a database server group and the second line defines a database server that is a member of the **srvB** server group:

```
srvB       group        -          -          i=83
svrB       ontlitcp     smoke      svcname    g=srvB
```

For more information about database server groups, refer to your *Administrator's Guide*.

## Time Specifications

When you set a frequency for replication in **cdr_define_repl()** or **cdr_modify_repl()**, state the time in the local time unless the **$TZ** environment variable is set. In the global catalog, all times are stored in Greenwich mean time (GMT).

Several API functions specify a time variable that has no function. You must, nevertheless, fill in a dummy value for the variable.

## Identifiers

Identifiers are the names of objects, such as database names, table names, and replicate names. For more information, refer to "Viewing Long Identifiers" on page 8-8.

## API Structures

The API functions documented in this chapter use the following structures, which are also documented in the following directories:

- **$INFORMIXDIR/incl/esql/cdrapi.h** (UNIX)
- **%INFORMIXDIR%\incl\esql\cdrapi.h** (Windows NT)

```
typedef struct plist
{
    char      serv[CDR_GL_NAMESIZE + 1];          /* server-group name */
    char      db[CDR_GL_NAMESIZE + 1];             /* database */
    char      table[CDR_GL_NAMESIZE + 1];          /* table */
    char      owner[CDR_GL_USERSIZE + 1];          /* owner */
    char      selectstmt[CDR_GL_SELECTSIZE + 1];   /* select stmt */
    uint4     partmode;                            /* participant mode */
    struct plist      *part_next;                  /* next participant */
} CDR_plist;

typedef struct freq
{
    uint2       hr;               /* hour 0-32767 */
    uint1       min;              /* minute 0-59 */
    uint1       type;             /* see frequency types below values */
    uint1       day;
    uint1       month;
} CDR_freq;
```

```
/* frequency types */
#define  FREQ_IMMED      1    /* Replicate all data immediately. This is the   */
                               /* default value.                                */
#define  FREQ_INTERVAL   2    /* Replicate data at the hour and minute         */
                               /* intervals specified by the hour and minute    */
                               /* members of the structure.                     */
#define  FREQ_TIME       3    /* Replicate data every day at the specified time */
#define  FREQ_DAYOFWEEK  4    /* Replicate data once a week at the time and day */
                               /* specified by the hour, minute, and day members */
                               /* of the structure, where 1=Monday, 2=Tuesday    */
                               /* 3=Wednesday, and so forth.                    */
#define  REQ_DAYOFMONTH  5    /* Replicate data once a month at the time and   */
                               /* day of the month specified by the hour,       */
                               /* minute, and day members of the structure,     */
                               /* where the day member contains a value between */
                               /* 1 and 31 or L designating the last day of the */
                               /* month                                        */



typedef struct conf
{
    uint1   method;                                  /* conflict method */
    char    storedproc[CDR_GL_SPSIZE + 1];           /* stored procedure */
    uint1   storedprocopt;             /* stored procedure optomization option */
} CDR_conf;

typedef struct repl_attrs
{
    uint4       flags;                  /* replicate attributes */
    time_t      begin;                  /* dummy variable */
    CDR_freq*   freq;                   /* freq tx's sent */
    uint4       threshold;              /* threshold (in bytes) */
    CDR_conf*   primary;                /* primary method*/
    CDR_conf*   secondary;              /* secondary method */
} CDR_repl_attrs;



typedef struct idlist
{
    char    id[CDR_GL_NAMESIZE+1];      /* identifier */
    struct  idlist*  id_next;           /* next identifier */
} CDR_idlist;
```

```
typedef struct q_type
{
    uint1    qtype;                     /*
                                         * Unused field, only present
                                         * for backward compatability
                                         */
    char     dbspace[CDR_GL_NAMESIZE+1]; /* dbspace for Q_STABLE */
} CDR_qtype;



typedef struct serv_attrs
{
    uint2        sflags;                    /* definition flags */
    uint2        idle;                      /* connection time-out (minutes) */
    CDR_qtype*   sendq;                     /* send queue type */
    CDR_qtype*   recvq;                     /* receive queue type */
    char         atsDir[CDR_DIRNAMESIZE+1];     /*  ATS spool dir */
    char         risDir[CDR_DIRNAMESIZE+1];     /*  RIS spool dir */
    char         syncServ[CDR_GL_NAMESIZE+1];   /* db server-group to
                                                   sync catalog with */
} CDR_serv_attrs;
```

## Severe-Error Table

The *severe-error table* is a repository for remote errors that the user does not see at runtime. The same information is available in the **syscdrerror** table in the **sysmaster** database.

### *Information Maintained in the Severe-Error Table*

Suppose you define a replicate between servers **g_server1** and **g_server3** while connected to **g_server1**. The replicate definition will succeed on **g_server1** even if the table specified in the SELECT statement is spelled differently on **g_server3** than on **g_server1**. However, the message log for server **g_server3** will display an error message and an appropriate error is written into the severe-error table.

### Retrieving Information from the Severe-Error Table

The severe-error table is replicated to all servers in the replication system, so the information is available from any server in your replication environment. However, *do not* attempt to examine this table directly. The **syscdr** database is reserved for internal use. To examine the error information, use the **syscdrerror** table of the **sysmaster** database.

To examine and manage the severe-error table, use the following tools:

- CLU option **cdr error**
- API functions:
  - **cdr_error_reviewed**()
  - **cdr_move_errortab**()
  - **cdr_prune_errors**()
  - **cdr_prune_single_error**()

## Enterprise Replication APIs

The following table lists the API functions for Enterprise Replication.

| API | Page |
|---|---|
| **cdr_connect()** | 12-10 |
| **cdr_define_group()** | 12-11 |
| **cdr_define_repl()** | 12-14 |
| **cdr_define_serv()** | 12-18 |
| **cdr_delete_group()** | 12-20 |
| **cdr_delete_repl()** | 12-21 |
| **cdr_delete_serv()** | 12-22 |
| **cdr_error_reviewed()** | 12-23 |
| **cdr_modify_group()** | 12-24 |

(1 of 2)

| API | Page |
|---|---|
| **cdr_modify_repl()** | 12-25 |
| **cdr_modify_replmode()** | 12-27 |
| **cdr_modify_serv()** | 12-28 |
| **cdr_modify_servmode()** | 12-29 |
| **cdr_move_errortab()** | 12-30 |
| **cdr_participate_group()** | 12-31 |
| **cdr_participate_repl()** | 12-32 |
| **cdr_prune_errors()** | 12-33 |
| **cdr_prune_single_error()** | 12-34 |
| **cdr_resume_group()** | 12-35 |
| **cdr_resume_repl()** | 12-36 |
| **cdr_resume_serv()** | 12-37 |
| **cdr_start_cdr()** | 12-38 |
| **cdr_start_group()** | 12-39 |
| **cdr_start_repl()** | 12-40 |
| **cdr_stop_cdr()** | 12-41 |
| **cdr_stop_group()** | 12-42 |
| **cdr_stop_repl()** | 12-43 |
| **cdr_suspend_group()** | 12-44 |
| **cdr_suspend_repl()** | 12-45 |
| **cdr_suspend_serv()** | 12-46 |


(2 of 2)

# cdr_connect()

The **cdr_connect()** function designates the origin for control and configuration.

***Important:*** *If you plan to call programs in addition to Enterprise Replication, you must manage your own connections.*

## Syntax

```
#include "cdrapi.h"

int
cdr_connect(dbserver)
    char      *dbserver;
```

*dbserver* is the name of the replication database server.

## Usage

The **cdr_connect()** function specifies the database server *dbserver* as the origin for all control and configuration calls. A connection is established to the database server *dbserver*. If *dbserver* is null, or if you do not invoke this call before making other API calls, the connection defaults to the database server that the **INFORMIXSERVER** environment variable specifies.

The database server specified must be defined in the global catalog. To define the database server for Enterprise Replication, use **cdr_define_serv().**

Unlike the other API commands that Enterprise Replication uses, the **cdr_connect()** function uses the database server name. All other API commands that reference a database server use the name of the database server group. For more information about database servers and database server group names, refer to "Database Server Groups" on page 12-3.

# cdr_define_group()

The **cdr_define_group()** function creates a replicate group in the global catalog.

## Syntax

```
#include "cdrapi.h"

int
cdr_define_group(group, attrs, repls, freq, which)
    char         *group;
    uint4         attrs;
    CDR_idlist   *repls;
    CDR_freq     *freq;
    void         *which;
```

*group* is the name of the replicate group.

*attrs* are the group attributes.

*repls* is the list of replicate names.

*freq* are the frequency attributes.

*which* if the call fails, this pointer references the argument that caused the error (see Appendix C).

## Usage

The **cdr_define_group** function defines a replicate group that includes the replicates listed in *repls*. The replicates specified must be defined over the same set of replication database servers and must all be in the same replication state. Any valid replicate can be defined as part of a replicate group providing the participant list is identical.

The group name must be no longer than 128 bytes and must start with a letter or an underscore.

All replicates must be in the same replication state. That is, all replicates must be active or all replicates must be inactive.

The *attrs* argument specifies attributes for the replicate group. The following attributes specify how the system processes data for the replicate group. To specify the attributes, set the appropriate flag bit in *attrs*.

| Flag Value | Description |
|---|---|
| A_NONSEQUENTIAL | This option configures Enterprise Replication to process all nonsequential groups in parallel. Use this option only when groups have disjoint data domains. The default is sequential processing of groups. |

The *freq* argument specifies the frequency with which data is replicated to database servers that participate in the replicate group. This defaults to the frequency of the participating replicates if the argument is null. The following frequency types can be specified by setting the type member of the frequency structure.

| Flag Value | Description |
|---|---|
| FREQ_IMMED | Replicate all data immediately. |
| FREQ_INTERVAL | Replicate data at the hour and minute interval specified by the hour and minute members of the structure. |
| FREQ_TIME | Replicate data every day at the time specified by the hour and minute members of the structure. |
| FREQ_DAYOFWEEK | Replicate data once a week at the time and day specified by the hour, minute, and day members of a structure where the day member contains a value between 1 and 7 where 1 = Monday, 2 = Tuesday, 3 = Wednesday, and so on. |
| FREQ_DAYOFMONTH | Replicate data once a month at the time of day of the month specified by the hour, minute, and day members of the structure where the day member contains a value between 1 and 31, or L, designating the last day of the month. |

If a replicate group is defined over a set of existing replicates whose frequency type is not immediate, data is replicated for the replicate group with respect to the earliest time that data was replicated for all replicates.

If you add a replicate that uses time-based frequency to an already existing group that uses the frequency type immediate, then all the data from the newly-added replicate is immediately sent to its targets even though the specified time has not arrived. This allows the newly-added replicate to assume the frequency attributes of the group with no pending data in its send queue. It also prevents Enterprise Replication from replicating data out of order.

Similarly, when you remove a replicate from the replicate group, the data queued for the entire group is immediately sent to the targets. The replicate that you removed from the group maintains the frequency attributes of the replicate group. That is, the replicate that you remove from the group inherits the frequency attributes of the replicate group.

After a replicate is defined as part of a replicate group, you must use replicate-group control commands for control and configuration. You cannot control the replicate individually until either the replicate is removed from the replicate group or the replicate group is deleted.

# cdr_define_repl()

The **cdr_define_repl()** function defines a replicate in the global catalog.

## Syntax

```
#include "cdrapi.h"

int
define_repl(repl_name, attrs, mflags, participants, which)
    char            *repl_name;
    CDR_repl_attrs  *attrs;
    uint4           mflags;
    CDR_plist       *participants;
    void            *which;
```

| | |
|---|---|
| *repl_name* | is the name of the replicate to define. |
| *attrs* | are the replicate attributes. |
| *mflags* | are the definition flags. |
| *participants* | are the participating database servers and associated attributes. |
| *which* | points to information if an error occurs. (See "Using the Stored Error Information" on page C-1.) |

## Usage

The **cdr_define_repl()** function defines the replicate *repl_name* with attributes *attrs* and participants *participants* in the global catalog. For definitions of the **CDR_repl_attrs** and **CDR_plist** structures, refer to "API Structures" on page 12-5. The replicate name can be no longer than 128 bytes and must start with a letter or an underscore.

### Attribute Values for CDR_repl_attrs

The *attrs* argument is a pointer to the **CDR_repl_attrs** structure, which specifies attributes for the replicate. These attributes specify how the system manages replication data for the replicate.

### The flags field of CDR_repl_attrs

The **flags** field of the **CDR_repl_attrs** structure can take the following values. The setting of the flags field influences the values in the primary and secondary fields. where applicable, set the appropriate related field.

The arguments of the **CDR_repl_attrs** structure specify the attributes of the replicate, as follows.

| Type of Value | Flags Field Macros | Description |
|---|---|---|
| Conflict resolution option | A_PRIMARY | The structure member ***attrs.primary*** specifies the primary resolution method for the replicate. The member ***attrs.primary.method*** specifies the conflict-resolution method. |
| Conflict resolution option | A_SECONDARY | The structure member ***attrs.secondary*** specifies the secondary resolution method for the replicate. The member ***attrs.secondary.method*** specifies the type. The member ***attrs.secondary*** is only valid if the primary method selected is time stamp. The only valid bit is STORED_PROCEDURE. |
| Conflict resolution scope | A_RSCOPE | Activates conflict resolution on a row-by-row basis. This is mutually exclusive with A_TSCOPE. |
| Conflict resolution scope | A_TSCOPE | Activates conflict resolution on a transaction basis. This is mutually exclusive with A_RSCOPE. |
| Diagnostic information | A_RIS | Activates row-information spooling for replicate row data that fails conflict resolution or encounters replication-order problems. (For more information, see "Row-Information Spooling" on page 10-9.) |
| Diagnostic information | A_ATS | Activates aborted-transaction spooling for replicate transactions that fail to be applied to the target database. (For more information, see "Aborted-Transaction Spooling" on page 10-3.) |

(1 of 2)

| Type of Value | Flags Field Macros | Description |
|---|---|---|
| Data format | A_CANONICAL | All replication data for the replicate is delivered in machine-independent form. This machine-independent format is required for replicates defined over a heterogeneous collection of database servers where the replication data is of type *float* or *double*. |
| Trigger activation | A_TRIGGER | Triggers should be fired for this replicate. The default fires no triggers for replicated updates. |
| Frequency specification | A_FREQ | Sets the frequency of data replication to the interval specified by **attrs.freq**.<br>See the A_BEGIN flag bit for discussion of time zones. The default is to distribute replication data immediately. |
| Owner value for permission checking | A_USETABOWNER | Specifies that replicated data should be applied on the target as the owner of the table. Permission checking for triggers should use the owner of the table instead of user **informix**. |

(2 of 2)

If the **R_PARTICIPANTS** flag bit is set in **mflags**, the system expects the **participants** argument to be present. This argument specifies the *replicate participants* for the replicate. Participant information is stored in the **CDR_plist** structure. The **selectstmt** member of that structure specifies the criteria for replication. The following requirements are necessary for the statement:

- The SELECT clause can reference only columns from the table being replicated.
- The SELECT clause must contain at least the primary key.
- The SELECT clause can contain only limited user-defined types. For information about using user-defined data types, refer to "Replicating User-Defined Data Types" on page 5-15.
- The WHERE clause of the SELECT statement cannot contain aggregates or constants.

### Attribute Values for CDR_conf

The **CDR_repl_attrs** includes the **CDR_conf** structure. The arguments of the **CDR_conf** structure specify the conflict resolution method, as follows.

| Type of Value | Value of flags Field | Description |
|---|---|---|
| Resolution type | R_IGNORE | Updates are applied as received; no update collisions are detected or processed. Replicates using this type of resolution do not use any special Enterprise Replication columns.<br>This value is mutually exclusive with R_TIMESTAMP and R_STOREDPROC. |
| Resolution type | R_TIMESTAMP | The replication time stamp determines which update will prevail.<br>This value is mutually exclusive with R_IGNORE and R_TIMESTAMP. |
| Resolution type | R_STOREDPROC | A stored procedure that the structure member ***attrs.primary.storedproc*** specifies is used to resolve the conflict. If you set this flag, you must allocate a **CDR_conf** structure and set the **primary** field.<br>This value is mutually exclusive with R_IGNORE and R_TIMESTAMP.<br>If you set R_STOREDPROC, you can set the stored-procedure behavior to R_STOREDPROC_OPTM or R_STOREDPROC_EXEC. |
| Stored procedure behavior | R_STOREDPROC_OPTM | Optimize behavior: the replicated row is accepted and the stored procedure is not called, even when conflict is detected, if the replicated row is from the same database server that last updated the local row on the target table, and the replicated row has a time stamp greater than or equal to the local row.<br>This value is mutually exclusive with R_STOREDPROC_EXEC. |
| Stored procedure behavior | R_STOREDPROC_EXEC | Execute behavior: the stored procedure defined for the replicate is always called when conflict is detected.<br>This value is mutually exclusive with R_STOREDPROC_OPTM. |

# cdr_define_serv()

The **cdr_define_serv()** function registers a database server as a replication server.

## Syntax

```
#include "cdrapi.h"

int
cdr_define_serv(server,attrs)
    char              *server;
    CDR_serv_attrs    *attrs;
```

*server* is the name of a database server group.

*attrs* are the database server attributes.

## Usage

The **cdr_define_serv()** function registers the database server specified by the database server group *server* as a replication server with the attributes *attrs*. A *replication server* is simply a database server that includes replication information. A database server cannot participate in Enterprise Replication until it has been registered as a replication server.

The **cdr_define_serv()** function stores information about the replication server in the global catalog in the **syscdr** database. If the **syscdr** database does not already exist, the **cdr_define_serv()** function creates it.

***Tip:*** *Do not try to examine the **syscdr** database. The information is stored in a format that is suitable only for use by the database server. For information about replication servers, examine the **syscdrservers** table in the **sysmaster** database. For a description of the Enterprise Replication tables in the **sysmaster** database, refer to Appendix B, "SMI Tables for Enterprise Replication."*

The database server group must be defined in the **sqlhosts** file (UNIX) or the SQLHOSTS registry key (Windows NT). For more information, refer to "Database Server Groups" on page 12-3.

The *attrs* argument specifies attributes for the database server. To specify attributes, set the appropriate flag bit in **attrs.sflags** and use the supplied structure member, as the following table shows.

| Attribute | Description |
|---|---|
| A_ATS | Use **attrs.ats_dir** as the directory for ATS files. For more information, see Chapter 10. |
| A_IDLE | All database servers maintain a separate connection to all other database servers participating in replication exclusively for the transmission of replication data. Because the connection consumes system resources, an idle time-out provides a way to optimize infrequent connections. The connection will be automatically re-established when data is queued for the database server. The idle time-out is specified in minutes by **attrs.timeout**. |
| A_INIT | Enterprise Replication is initialized on the database server that is defined. |
| A_LEAF | Define this replication server as a leaf server. A_ROOT, A_NONROOT, and A_LEAF are mutually exclusive. |
| A_NONROOT | Define this replication server as a nonroot server. |
| A_RECVQ | Use **attrs.recvq** for the receive-queue dbspace. The receive queues is assigned to a particular dbspace. The receive-queue dbspace cannot be a temporary dbspace. The path for the receive-queue dbspace must be no longer than 128 bytes. |
| A_ROOT | Define this replication server as a root server. |
| A_RIS | Use **attrs.ris_dir** as the directory for RIS files. For more information, see Chapter 10. |
| A_SENDQ | Use **attrs.sendq** for the send-queue dbspace. The send queue is assigned to a particular dbspace. The send-queue dbspace cannot be a temporary dbspace. The path for the send-queue dbspace must be no longer than 128 bytes. |
| A_SYNC | Use **attrs.syncServ** as the database server for catalog synchronization when the database server is initialized. This option is only available when the A_INIT flag has been set. |

# cdr_delete_group()

The **cdr_delete_group()** function deletes a replicate group.

## Syntax

```
#include "cdrapi.h"

int
cdr_delete_group(group, time)
    char     *group;
    time_t   time;          /* unused */
```

*group* is the replicate group name.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

## Usage

The **cdr_delete_group()** function deletes the replicate group *group* from the global catalog.

The **cdr_delete_group()** function does not affect the replicates or associated data. When a replicate group is deleted, the state of individual replicates within the group is unchanged.

# cdr_delete_repl()

The **cdr_delete_repl()** function deletes a replicate from the global catalog.

## Syntax

```
#include "cdrapi.h"

int
cdr_delete_repl(repl, time)
    char      *repl;
    time_t    time;          /* unused */
```

*repl* is the name of the replicate to delete.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

## Usage

The **cdr_delete_repl()** function deletes the replicate *repl* from the global catalog. All replication data for the replicate is purged from the send and queues at all participating database servers.

If *repl* is contained in a replicate group, the call fails. You must remove the replicate from the group before you can delete it. To remove a replicate from a group, see "cdr_participate_group()" on page 12-31.

# cdr_delete_serv()

The **cdr_delete_serv()** function deletes a database server from the global catalog.

## Syntax

```
#include "cdrapi.h"

int
cdr_delete_serv(server)
    char    *server;
```

*server* is the name of a database server group.

## Usage

The **cdr_delete_serv()** function deletes the database server specified by the database server group *server* from the global catalog. The database server is removed from all participating replicates. All replication data is purged from the send queues for the database server specified. Enterprise Replication removes the global catalog from the replication server *server* and shuts down on that server.

Enterprise Replication does not broadcast the **cdr_delete_serv**() call to other replication servers. Therefore, the call needs to be issued twice, as follows:

1. Connect to the replication server that needs to be deleted (**serverA**) and issue **cdr_delete_serv()** to purge its catalog and delete all queued data.
2. Connect to another replication server (**serverB**) to purge the entry of the deleted server (**serverA**) from the catalog of **serverB**. This update to the catalog of **serverB** is then replicated to all other replication servers.

# cdr_error_reviewed()

The **cdr_error_reviewed()** function marks a specified error as reviewed.

## Syntax

```
#include "cdrapi.h"

int
cdr_error_reviewed (orig_server, remote_seqnum)
    char       *orig_server;
    long       remote_seqnum;
```

*orig_server* is the name of a database server group.

*remote_seqnum* is the remote sequence-number value for the error that is to be updated. Combined with the originating database server name, remote_seqnum provides a unique key for every error.

## Usage

The **cdr_error_reviewed()** function marks an error in the severe-error table as having been reviewed. The *orig_server* specifies the database server group that includes the database server where the error originated.

When an error is *reviewed*, it means that someone has resolved the reason for a given error. The review state is used only to track and determine which errors to delete with **cdr_prune_errors()**. The originating database server name and the sequence number from that database server are used to uniquely specify the error to be marked. The specified error is updated on all database servers participating in replication.

# cdr_modify_group()

The **cdr_modify_group()** function modifies the attributes of a replicate group.

## Syntax

```
#include "cdrapi.h"

int
cdr_modify_group(group, attrs, freq)
    char        *group;
    uint4       attrs;
    CDR_freq    *freq;
```

*group* is the replicate group name.

*attrs* are the replicate group attributes.

*freq* are the frequency attributes.

## Usage

The **cdr_modify_group** function modifies the group *group* with attributes *attrs*. The *attrs* argument specifies which attributes to modify. For a description of the attributes, see "cdr_define_group()" on page 12-11. The *freq* argument can be null; in such cases the frequency attributes of the replicate group are not modified.

You cannot remove replicates from a group when the group is suspended. Also, you cannot add a suspended replicate to a group, even if the replicate group is also suspended.

To add or remove replicates from a replicate group, use "cdr_participate_group()" on page 12-31.

# cdr_modify_repl()

The **cdr_modify_repl()** function modifies replicate attributes.

## Syntax

```
#include "cdrapi.h"

int
cdr_modify_repl(repl_name,attrs)
    char              *repl;
    CDR_repl_attrs    *attrs;
```

*repl_name* is the replicate name.

*attrs* are the frequency attributes.

## Usage

The **cdr_modify_repl** function modifies the replicate *repl* with the replicate attributes *attrs*. The *attrs* argument specifies the new attributes for the replicate. For the definition of the **CDR_repl_attrs** structure, refer to "API Structures" on page 12-5.

### *Attributes Defined cdr_define_repl() and cdr_modify_repl()*

You can use the following attributes with both **cdr_define_repl()** and **cdr_modify_repl()**. For a description of these attributes, see "cdr_define_repl()" on page 12-14.

- A_ATS
- A_BEGIN
- A_CANONICAL
- A_FREQ
- A_PRIMARY
- A_RIS
- A_RSCOPE
- A_SECONDARY

- A_TRIGGER
- A_TSCOPE
- A_USETABOWNER

You cannot modify the **ignore** conflict rule to **timestamp** or **stored procedure**, nor can you modify **timestamp** or **stored procedure** to **ignore**. However, you *can* use A_PRIMARY and A_SECONDARY to modify the conflict resolution rule from **timestamp** to **stored procedure** and back again.

### Additional Attribute Values for cdr_modify_repl()

In addition to the attributes specified in **cdr_define_repl()**, you can use the following attribute values with **cdr_modify_repl()**.

| Type of Value | Value of flags Field | Description |
|---|---|---|
| Transaction spooling | A_NRIS | Deactivate row-information spooling for this replicate. |
| Transaction spooling | A_NATS | Deactivate aborted-transaction spooling for this replicate. |
| Triggers | A_NTRIGGER | Triggers should not be fired for this replicate. |
| Frequency | A_FREQ | Modifies the frequency according to the frequency in the attributes. |

# cdr_modify_replmode()

The **cdr_modify_replmode()** function changes the mode of participants in a replicate.

## Syntax

```
#include "cdrapi.h"

int
cdr_modify_replmode(repl, participants, which)
    char        *repl;
    CDR_plist   *participants
    void        *which;
```

*repl* is the replicate name.

*participants* is the list of participants.

*which* points to information if an error occurs. (See "Using the Stored Error Information" on page C-1.)

## Usage

The **cdr_modify_replmode** function modifies the modes of replicate participant *participants* in replicate *repl*. The *partmode* field of each participant structure sets the mode of the participant. For the definition of the **CDR_plist** structure, refer to "API Structures" on page 12-5.

For a description of the attributes in the **CDR_plist** structure, refer to "cdr_define_repl()" on page 12-14

If the value of *partmode* is A_RDONLYMODE, then the participant is in read-only state, where updates made to the replicated table on the database server for this participant are not replicated.

If the value of *partmode* is A_PRIMARYMODE, then the participant is in primary state, where updates made to the replicated table on the participant database server are replicated. A_PRIMARYMODE is the default mode.

# cdr_modify_serv()

The **cdr_modify_serv()** function modifies the Enterprise Replication attributes of a database server.

## Syntax

```
#include "cdrapi.h"

int
cdr_modify_serv(server,attrs)
    char       *server;
    CDR_serv_attrs  *attrs;
```

*server* is the name of a database server group.

*attrs* are the database server attributes to modify.

## Usage

The **cdr_modify_serv()** function modifies the attributes *attrs* of the database server specified by the database server group *server*. The same flag bits are supported as for **cdr_define_serv()**, except that you cannot modify the send queue (A_SENDQ) or the receive queue (A_RECVQ). For a description of attributes, see "cdr_define_serv()" on page 12-18.

# cdr_modify_servmode()

The **cdr_modify_servmode()** function changes the *mode* of participants for a database server.

## Syntax

```
#include "cdrapi.h"

int
cdr_modify_servmode(server, mode)
    char      *server;
    uint4     mode;
```

*server* is the name of the database server group that includes the database server whose mode should be changed.

*mode* is the mode to use for all the participants for this database server. It can be either A_RDONLYMODE or A_PRIMARYMODE.

## Usage

The **cdr_modify_servmode()** funtion changes the mode of all replicate participants for the specified database server. The *mode* argument determines the mode for all participants of the database server.

A_PRIMARYMODE mode sets the participants to primary mode.

A_RDONLYMODE mode sets the participants to read-only mode. For A_RDONLYMODE, participants of replicates with conflict resolution that are not set to ignore are unchanged.

Participants already in the designated mode are unaffected.

# cdr_move_errortab()

The **cdr_move_errortab()** function moves the error table to another dbspace.

## Syntax

```
#include "cdrapi.h"

int
cdr_move_errortab(dbspace, performonall)
    char      *dbspace;
    int       performonall;
```

*dbspace* is the name of the dbspace in which the error table is to be moved.

*performonall* is the flag that indicates whether this operation should be performed on all database servers or only the database server to which the user is connected.

## Usage

The **cdr_move_errortab()** function moves the severe-error table to a new dbspace. All errors that are currently stored in the table are moved to the new dbspace. If the **performonall** flag is nonzero, this operation is performed on all database servers involved in replication. If the **performonall** flag is zero, this function affects only the database server to which the user is connected.

For more information, refer to the "Severe-Error Table" on page 12-7.

# cdr_participate_group()

The **cdr_participate_group()** function modifies a replicate group by adding or deleting replicates.

## Syntax

```
#include "cdrapi.h"

int
cdr_participate_group(group, op, repls, which)
    char        *group;
    uint1       op;
    CDR_idlist  *repls;
    void        *which;
```

*group* is the name of the replicate group.

*op* is the operation A_ADD or A_REMOVE.

*repls* is the list of replicates.

*which* points to information if an error occurs. (See "Using the Stored Error Information" on page C-1.)

## Usage

The **cdr_participate_group()** function modifies the replicate group *group* by adding or removing replicates.

If the value of *op* is **A_ADD**, the function adds the replicates in the *repls* list to the group.

If the operation is type **A_REMOVE**, the function removes the replicates in the *repls* list from the group. If the function removes the last participant from the group, the function also removes the group from the global catalog.

Removing a replicate from a group does *not* destroy the replicate or change its state. The state of the replicate state remains the same as the state of the replicate group from which the replicate was removed.

# cdr_participate_repl()

The **cdr_participate_repl()** function adds or removes participants from a replicate.

## Syntax

```
#include "cdrapi.h"

int
cdr_participate_repl(repl, participants, op, which)
    char          *repl;
    uint1         op;
    CDR_plist     *participants;
    void          *which;
```

*repl* is the replicate name.

*participants* is the list of participants.

*op* is the operation type: **A_ADD** or **A_REMOVE**.

*which* if the call fails, this pointer references information about the error. (See "Using the Stored Error Information" on page C-1.)

## Usage

The **cdr_participate_repl()** function modifies the replicate *repl* by adding or removing the participants in the *participants* list.

If the value of *op* is **A_ADD**, the function adds the participants in the *participants* list to the replicate. The replication state is initialized to *inactive*.

If the value of *op* is **A_REMOVE**, the function removes the participants in the *participants* list from the replicate. The *selectstmt* member of the **CDR_plist** structure is not required and is ignored if it is present. For the definition of the **CDR_plist** structure, refer to "API Structures" on page 12-5

# cdr_prune_errors()

The **cdr_prune_errors()** function prunes errors from the severe-error table based on time or time range.

## Syntax

```
#include "cdrapi.h"
#include "datetime.h"
#include "decimal.h"

int
cdr_prune_errors(uppertime, lowertime, reviewed_only)
    dtime_t         *uppertime;
    dtime_t         *lowertime;
    int             reviewed_only;
```

*uppertime* is the upper (most recent) time limit for deletion of errors. This value must be specified.

*lowertime* is the lower (earliest) time limit for deletion of errors. If a null is passed, all errors earlier than uppertime are deleted.

*reviewed_only* is a flag that specifies which errors to delete.

## Usage

The **cdr_prune_errors()** function deletes errors from the severe-error table on all database servers involved in replication for a time range, or for all errors before a specified time.

If *lowertime* is null, the function deletes all errors that occurred at or before *uppertime*. If both time values are specified, then errors that occurred within the time range are deleted.

If the *reviewed_only* flag is nonzero, only errors that match the specified time criteria *and* that have been reviewed are deleted.

For information about the severe-error table, refer to "Severe-Error Table" on page 12-7.

# cdr_prune_single_error()

The **cdr_prune_single_error()** function prunes a single error from the severe-error table.

## Syntax

```
#include "cdrapi.h"

int
cdr_prune_single_error(origserver, remote_seqnum)
    char   *origserver;
    long   remote_seqnum;
```

*origserver* is the name of the database server group that includes the database server from which the error originated.

*remote_seqnum* is the remote sequence number value for the error that should be deleted. Combined with origserver, this number provides a unique key for every error.

## Usage

The **cdr_prune_single_error** function deletes a single error from the severe-error table. It uses the name of the originating server group and the sequence number from that database server to uniquely specify the error to be deleted. The specified error is deleted from all database servers that participate in replication.

For more information about the severe-error table, refer to "Severe-Error Table" on page 12-7.

# cdr_resume_group()

The **cdr_resume_group()** function resumes delivery of replication data for replicates defined in a replicate group. This function is valid only on a group that has been suspended.

## Syntax

```
#include "cdrapi.h"

int
cdr_resume_group(group,time)
    char     *group;
    time_t   time;       /* unused */
```

*group* is the replicate group name.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

## Usage

The **cdr_resume_group()** function causes the replicates contained in the replicate group *group* to enter the active state (that is, to capture and send replication data) for all participants.

# cdr_resume_repl()

The **cdr_resume_repl()** function resumes delivery of replication data.

## Syntax

```
#include "cdrapi.h"

int
cdr_resume_repl(repl, time)
    char     *repl;
    time_t   time;        /* unused *
```

*repl* is the replicate name.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

## Usage

The replicate *repl* enters the active state (that is, to capture and send replication data) for all participants.

If *repl* is contained in a replicate group, the call fails. To resume delivery of replication data to a replicate group, see "cdr_resume_group()" on page 12-35.

# cdr_resume_serv()

The **cdr_resume_serv()** function resumes delivery of replication data to a database server.

## Syntax

```
#include "cdrapi.h"

int
cdr_resume_serv(db_server, servers, time, which)
    char          *db_server;
    CDR_idlist    *servers;
    time_t        time;       /* unused *
    void          *which;
```

*db_server* is the name of the database server group that includes the database server that should resume replication.

*servers* is an optional list of database server group names.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

*which* if the call fails, this pointer references information about the error. (See "Using the Stored Error Information" on page C-1.)

## Usage

The **cdr_resume_serv()** function causes delivery of replication data (including queued data) to resume to the database server *db_server* from all servers included in the server-group list *servers*. If *servers* is null, then replication resumes on all database servers.

# cdr_start_cdr()

The **cdr_start_cdr()** function restarts Enterprise Replication processing. This function is valid only after a **cdr_stop()**.

## Syntax

```
#include "cdrapi.h"

int
cdr_start_cdr()
```

## Usage

Use the **cdr_start_cdr()** function to restart Enterprise Replication after it is stopped with **cdr_stop_cdr()**. When you issue **cdr_start_cdr()**, Enterprise Replication threads start and continue from the point at which they stopped.

Enterprise Replication resumes evaluation of the logical log (if required for the instance of Enterprise Replication) at the *replay* position. The replay position is the position where Enterprise Replication stops evaluating the logical log when **cdr_stop_cdr()** is executed. If the evaluation process is running and the logical log ID for the replay position no longer exists when Enterprise Replication is started, then the restart partially fails (the database server log contains an error message stating that the replay position is invalid). If the restart partially fails, no database updates performed on the local database server are replicated.

***Warning:*** *Informix recommends that you issue **cdr_start_cdr()** and **cdr_stop_cdr()** with extreme caution. Use these functions during a period of little or no database activity and keep the Enterprise Replication downtime to a minimum. Transactions that occur during this period are not replicated, hence tables can become unsynchronized.*

# cdr_start_group()

The **cdr_start_group()** function starts the capture and transmittal of replication transactions for replicates contained in a replicate group.

## Syntax

```
#include "cdrapi.h"

int
cdr_start_group(group, servers, time, which)
    char          *group;
    CDR_idlist    *servers;
    time_t        time;
    void          *which;
```

*group* is the replicate group name.

*servers* is an optional list of database server groups for which replication should begin.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

*which* if the call fails, this pointer references information about the error. (See "Using the Stored Error Information" on page C-1.)

## Usage

The **cdr_start_group()** function causes the replicates defined in the replicate group *group* to enter the active state (capture-send) on the database servers in the *servers* list. If the *servers* list is null, then the specified replicate group enters the active state on all database servers.

# cdr_start_repl()

The **cdr_start_repl()** function starts the capture and transmittal of data from a replicate.

## Syntax

```
#include "cdrapi.h"

int
cdr_start_repl(repl, servers, time, which)
    char          *repl;
    CDR_idlist    *servers;
    time_t        time;
    void          *which;
```

*repl* is the replicate name.

*servers* is an optional list of database server groups for which replication should begin.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

*which* if the call fails, this pointer references information about the error. (See "Using the Stored Error Information" on page C-1.)

## Usage

The **cdr_start_repl()** function causes the replicate *repl* to enter the active state (data is captured and sent) on the database servers in *servers* list. If the *servers* list is null, then the specified replicates enter the active state on all database servers.

To enter the active state:

- synchronize active participants.
- begin to capture data for replication on the participants specified.

If **repl** is contained in a replicate group, the call fails. To start replication on a group, see "cdr_start_group()" on page 12-39.

# cdr_stop_cdr()

The **cdr_stop_cdr()** function stops Enterprise Replication processing.

## Syntax

```
#include "cdrapi.h"

int
cdr_stop_cdr()
```

## Usage

In most situations, Enterprise Replication starts when **cdr_define_serv()** is first executed from a database server. The Enterprise Replication threads remain running until the database server is shut down, or until the local database server is deleted with the **cdr_delete_serv()** call. If you shut down the database server while Enterprise Replication is running, Enterprise Replication restarts when you restart the database server.

Under rare conditions, users might want to temporarily stop the Enterprise Replication threads without stopping the database server. The **cdr_stop_cdr()** function shut downs all Enterprise Replication threads in an orderly manner. When **cdr_stop_cdr()** completes the shutdown, it writes the message `CDR: shutdown complete` in the database server log file.

After you issue a **cdr_stop_cdr()** function, Enterprise Replication threads remain stopped even if the database server is stopped and restarted. To restart Enterprise Replication, see "cdr_start_cdr()" on page 12-38.

***Warning:*** *Informix recommends that you issue* ***cdr_start_cdr()*** *and* ***cdr_stop_cdr()*** *with extreme caution. Use these functions during a period of little or no database activity and keep the Enterprise Replication downtime to a minimum. Transactions that occur during this period are not replicated and hence tables can become unsynchronized.*

# cdr_stop_group()

The **cdr_stop_group()** function stops the capture and transmittal of data from replicates contained in a replicate group.

## Syntax

```
#include "cdrapi.h"

int
cdr_stop_group(group, servers, time, which)
    char          *group;
    CDR_idlist    *servers;
    time_t        time;
    void          *which;
```

*group* is the replicate group name.

*servers* is an optional list of database server groups for which the replication should be stopped.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

*which* if the call fails, this pointer references information about the error. (See "Using the Stored Error Information" on page C-1.)

## Usage

All replicate data defined in the replicate group *group* enters the *inactive* state *(*no capture, no send*)* on the database servers in the *servers* list. If the *servers* list is null, replication stops for all replicates in the replicate group.

# cdr_stop_repl()

The **cdr_stop_repl()** function stops the capture and transmittal of data for a replicate.

## Syntax

```
#include "cdrapi.h"

int
cdr_stop_repl(repl, servers, time, which)
    char          *repl;
    CDR_idlist    *servers;
    time_t        time;
    void          *which;
```

*repl* is the replicate name.

*servers* is an optional list of database-server groups that include the database for which the replication should be stopped.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

*which* if the call fails, this pointer references information about the error. (See "Using the Stored Error Information" on page C-1.)

## Usage

The **cdr_stop_repl()** function causes replicate *repl* to enter the inactive state (no data capture, no send) on the database servers in the *servers* list. If the *servers* list is null, then replication stops for all database servers.

If **repl** is contained in a replicate group, the call fails. To stop data transmittal for a replicate group, see "cdr_stop_group()" on page 12-42.

# cdr_suspend_group()

The **cdr_suspend_group()** function suspends delivery of replication data for replicates contained in a replicate group.

## Syntax

```
#include "cdrapi.h"

int
cdr_suspend_group(group, time)
    char     *group;
    time_t   time;          /* not used */
```

*group* is the replicate group name.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

## Usage

The **cdr_suspend_group()** function causes the replicates contained in the replicate group *group* to enter the *suspend* state (that is, do not capture or send replication data*)* for all participants.

***Warning:*** *When a group is suspended, Enterprise Replication holds the replication data in the send queue until the group is resumed. If lots of data is generated for the group while it is suspended, the send queue space can fill, causing data to be lost*

***Warning:*** *Enterprise Replication does not synchronize transactions if a replicate is suspended. For example, a transaction that updates tables X and Y will be split if replication for table X is suspended.*

# cdr_suspend_repl()

The **cdr_suspend_repl()** function suspends delivery of data from a replicate.

## Syntax

```
#include "cdrapi.h"

int
cdr_suspend_repl(repl, time)
    char      *repl;
    time_t    time;          /* not used */
```

*repl* is the replicate name.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

## Usage

The replicate *repl* enters the *suspend* state (that is, do not capture or send replication data) for all participants.

If *repl* is contained in a replicate group, the call fails. To suspend replication on a replicate group, see "cdr_suspend_group()" on page 12-44.

***Warning:*** *When a replicate is suspended, Enterprise Replication holds the replication data in the send queue until the group is resumed. If lots of data is generated for this replicate while it is suspended, the send queue space can fill, causing data to be lost.*

***Warning:*** *Enterprise Replication does not synchronize transactions if a replicate is suspended. For example, a transaction that updates tables X and Y will be split if replication for table X is suspended.*

# cdr_suspend_serv()

The **cdr_suspend_serv()** function suspends the delivery of replication data to a database server.

## Syntax

```
#include "cdrapi.h"

int
cdr_suspend_serv(server, servers, time, which)
    char          *server;
    CDR_idlist    *servers;
    time_t        time;          /* not used */
    void          *which;
```

*server* is the database server name to suspend.

*servers* is an optional list of database server groups that include the database servers on which to suspend replication.

*time* is not used but cannot be omitted. Use `(time_t)0` for that argument.

*which* if the call fails, this pointer references information about the error. (See "Using the Stored Error Information" on page C-1.)

## Usage

The **cdr_suspend_serv** function suspends delivery of replication data to the database server specified by *server* from all database servers specified by the *servers* list. Replication data continues to flow for all other database servers.

If the *servers* list is null, delivery of replication data is suspended for all database servers. None of the replication servers will send data to the suspended server.

***Tip:*** *Unlike **cdr_suspend_repl()**, the replicate and/or replicate group status is irrelevant.*

## Example

In the following example, the five database server groups are **g_s1** through **g_s5**. The database server **s1** is the synchronization server.

```
cdr_connect(s1);    /* could be any of the five servers */
cdr_suspend_serv(g_s3, g_s1 g_s2 g_s4 g_s5, (time_t)0, err);
```

The preceding two lines cause replicated data to immediately stop flowing to **s3** from **s1**, **s2**, **s4** and **s5**.

The corresponding command in the CLU is:

```
cdr suspend serv -c s1 g_s3 g_s1 g_s2 g_s4 g_s5
```

# Appendixes

Section IV


**Appendix A** **Configuration Parameters**

**Appendix B** **SMI Tables for Enterprise Replication**

**Appendix C** **Return Codes**

**Appendix D** **Fine-Tuning Enterprise Replication**

**Appendix E** **SQLHOSTS Registry Key**

**Appendix F** **Enterprise Replication onstat -g Options**

Appendix

# A

# Configuration Parameters

The database server configuration file ($ONCONFIG) includes the following configuration parameters that affect the behavior of Enterprise Replication:

- CDR_DSLOCKWAIT
- CDR_EVALTHREADS
- CDR_LOGDELTA
- CDR_NIFCOMPRESS
- CDR_NIFRETRY
- CDR_QUEUEMEM

If you use both DBSERVERNAME and DBSERVERALIASES, DBSERVERNAME should refer to the network connection and not to a shared-memory connection. For information about database server aliases, refer to the *Administrator's Guide for Informix Dynamic Server 2000*.

# CDR_DSLOCKWAIT

*value* 5

*units* seconds

*takes effect* when shared memory is initialized

The CDR_DSLOCKWAIT configuration parameter specifies the number of seconds the data synchronization component waits for database locks to be released. The CDR_DSLOCKWAIT parameter behaves similarly to the SET LOCK MODE statement: while the SET LOCK MODE is set by the end user application, CDR_DSLOCKWAIT is used by Enterprise Replication while applying data at the target database. This parameter is useful in conditions where different sources require locks on the replicated table. These sources could be a replicated transaction from another server or a local application operating on that table.

Transactions that receive updates and deletes from another server in the replicate can abort because of locking problems. The Datasync (data synchronization) component of Enterprise Replication always does an index read on the replicated table. In order to avoid deadlock, the application using the table should also use index read. (When a database server inserts data into a table, it locks the index, which could potentially result in a collision with another database server.)

# CDR_EVALTHREADS

*value* 1,2

*units* evaluators

*range of values* 1 to 129

*takes effect* when shared memory is initialized

The CDR_EVALTHREADS configuration parameter specifies the number of grouper evaluator threads to create when Enterprise Replication starts and enables parallelism when Enterprise Replication evaluates transactions from the logical log. The format is:

```
(per-cpu-vp,additional)
```

The following table provides four examples of CDR_EVALTHREADS.

| Number of Threads | Explanation | Example |
|---|---|---|
| 1,2 | 1 evaluator thread per CPU VP, plus 2 | For a 3 CPU VP server: (3 * 1) + 2 = 5 |
| 2 | 2 evaluator threads per CPU VP | For a 3 CPU VP server: (3 * 2) = 6 |
| 2,0 | 2 evaluator threads per CPU VP | For a 3 CPU VP server: (3* 2)+0 = 6 |
| 0,4 | 4 evaluator threads for any database server | For a 3 CPU VP server: (3 * 0) +4=4 |

# CDR_LOGDELTA

*default value* 30

*range of values* 0 to 100

*takes effect* when Enterprise Replication is initialized

The CDR_LOGDELTA configuration parameter specifies the percent of the space reserved for logical logs that can be devoted to memory for the send and receive queues.

If the evaluation of transactions for replication is much slower than the actual rate of transactions, the evaluation can fall considerably behind. In that case, Enterprise Replication blocks out further transactions to let the transaction evaluation catch up. If your log size is small, you can increase CDR_LOGDELTA to prevent the database server from overcommitting queue memory during this catch up phase.

You can also use CDR_QUEUEMEM to limit the memory that the database server uses for the send and receive queues. The database server uses the lower of the values derived from CDR_QUEUEMEM and CDR_LOGDELTA. For information about CDR_QUEUEMEM, refer to "CDR_QUEUEMEM" on page A-7.

# CDR_NIFCOMPRESS

*default value* 0

*range of values*
- -1 do not compress
- 0 compress only if the target server expects compression
- 1 - 9 increasing levels of compression

*takes effect* when Enterprise Replication is initialized

The CDR_NIFCOMPRESS (network interface compression) configuration parameter specifies the level of compression that the database server uses before sending data from the source database server to the target database server. Network compression saves network bandwidth over slow links but uses more CPU to compress and decompress the data.

The values have the following meanings.

| Value | Meaning |
|---|---|
| -1 | The source database server never compresses the data, regardless of whether or not the target site uses compression. |
| 0 | The source database server compresses the data only if the target database server expects compressed data. |
| 1 | The database server performs a minimum amount of compression. |
| 9 | The database server performs the maximum possible compression. |

When Enterprise Replication is defined between two database servers, the CDR_NIFCOMPRESS values of the two servers are compared and changed to the higher compression values.

The compression values determine how much memory can be used to store information while compressing, as follows:

```
    0 = no additional memory
    1 = 128k + 1k = 129k
    2 = 128k + 2k = 130k
... 6 = 128k + 32k  = 160k
... 8 = 128k + 128k = 256k
    9 = 128k + 256k = 384k
```

Higher levels of CDR_NIFCOMPRES cause greater compression.

Different sites can have different levels. For example, Figure A-1 shows a set of three root servers connected with LAN and a nonroot server connected over a modem. The CDR_NIFCOMPRES configuration parameter is set so that connections between A , B, and C use no compression. The connection from C to D uses level 6.

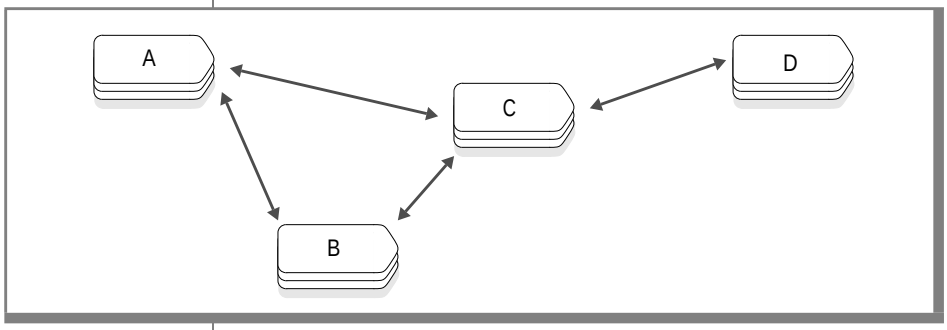



***Figure A-1***
*Database Servers with Different Compression Levels*

## CDR_NIFRETRY

| | |
|---|---|
| *default value* | 300 |
| *units* | seconds |
| *range of values* | any reasonable value |
| *takes effect* | when shared memory is initialized |

The CDR_NIFRETRY configuration parameter specifies the maximum time between retries when the database server is trying to make a connection for Enterprise Replication. If the value is 0, the database server does not retry the connection.

The first retry happens after 15 seconds and the next after 30 seconds. The retry interval keeps increasing, doubling the previous retry interval until the CDR_NIFRETRY is reached.

Retry happens only if the connection goes down abnormally, as in a power failure. A graceful shutdown of a server does not cause a retry.

## CDR_QUEUEMEM

| | |
|---|---|
| *value* | 4096 |
| *units* | kilobytes |
| *range of values* | > = 500 |
| *takes effect* | when shared memory is initialized |

The CDR_QUEUEMEM configuration parameter specifies the maximum amount of memory that is used for the send and receive queues. If your logical logs are large, you can use CDR_QUEUEMEM to limit the amount of memory devoted to queues.

You can also use CDR_LOGDELTA to limit the memory that the database server uses for the send and receive queues. The database server uses the lower of the values derived from CDR_QUEUEMEM and CDR_LOGDELTA. For information about CDR_LOGDELTA, see "CDR_LOGDELTA" on page A-4.

The maximum memory used for queued elements on a replication server is the total number of database servers involved in replication, multiplied by the value of CDR_QUEUEMEM.

When you increase the value of CDR_QUEUEMEM, you reduce the number of elements that must be written to disk, which can eliminate I/O overhead. Therefore, if elements are frequently stored on disk, increase the value of CDR_QUEUEMEM. Tune the value of CDR_QUEUEMEM for the amount of memory available on your computer.

Appendix

# B

# SMI Tables for Enterprise Replication

The system-monitoring interface (SMI) tables in the **sysmaster** database provide information about the state of the database server. Enterprise Replication uses the following SMI tables to access information about database servers declared to Enterprise Replication.

| Table | Description | Reference |
|---|---|---|
| **syscdrack_buf** | Buffers for the acknowledge queue | page B-5 |
| **syscdrack_txn** | Transactions in the acknowledge queue | page B-6 |
| **syscdrctrl_buf** | Buffers for the control queue | page B-6 |
| **syscdrctrl_txn** | Transactions in the control queue | page B-6 |
| **syscdrerror** | Enterprise Replication error information | page B-7 |
| **syscdrgrp** | Replicate group information | page B-7 |
| **syscdrgrppart** | Detailed information about replicate groups | page B-8 |
| **syscdrpart** | Participant information | page B-8 |
| **syscdrprog** | Contents of the Enterprise Replication progress tables | page B-9 |
| **syscdrq** | Queued data information (brief) | page B-10 |
| **syscdrqueued** | Queued data information (detailed) | page B-11 |

| Table | Description | Reference |
|---|---|---|
| **syscdrrecv_buf** | Buffers in the receive queue | page B-11 |
| **syscdrrecv_txn** | Transactions in the receive queue | page B-11 |
| **syscdrrepl** | Replicate information | page B-12 |
| **syscdrs** | Server information | page B-13 |
| **syscdrsend_buf** | Buffers in the send queue | page B-14 |
| **syscdrsend_txn** | Transactions in the send queue | page B-15 |
| **syscdrserver** | Database server information | page B-15 |
| **syscdrsync_buf** | Buffers in the synchronization queue | page B-16 |
| **syscdrsync_txn** | Transactions in the synchronization queue | page B-16 |
| **syscdrtx** | Transaction information | page B-17 |
| **syscdrtxproc** | Transaction processes information | page B-17 |

(2 of 2)

# Enterprise Replication Queues

One group of **sysmaster** tables shows information about Enterprise Replication queues. The **sysmaster** database reports the status of these queues in the tables that have the suffixes **_buf** and **_txn**.

The name of each table that describes an Enterprise Replication queue is composed of the following three pieces:

- An initial part, **syscdr**, that indicates that the table describes Enterprise Replication
- An abbreviation that indicates which queue the table describes
- A suffix, either **_buf** or **_txn**that specifies whether the table includes buffers or transactions

Selecting from these tables provides information about the contents of each queue. For example, the following SELECT statement returns a list of all transactions queued on the send queue:

```
SELECT * FROM syscdrsend_txn
```

The following example returns a list of all transactions queued on the in-memory send queue and returns the number of buffers and the size of each buffer for each transaction on the send queue:

```
SELECT cbkeyserverid,cbkeyid,cbkeypos,count(*),sum(cbsize)
    from syscdrsend_buf
    group by cbkeyserverid, cbkeyid, cbkeypos
    order by cbkeyserverid, cbkeyid, cbkeypos
```

All queues are present on all the replication servers, regardless of whether the replication server is a source or a target for a particular transaction. Some of the queues are always empty. For instance, the send queue on a target-only server is always empty.

Each queue is two-dimensional. Every queue has a list of transactions headers. Each transaction header in turn has a list of buffers that belong to that transaction.

## Columns of the Transaction Tables

All the tables whose names end with **_txn** have the same columns and the same column definitions. The information in the tables represents *only* transactions in memory and not those spooled to disk.

The **ctstamp1** and **ctstamp2** columns combine to form the primary key for these tables.

| Column | Type | Description |
|---|---|---|
| **ctkeyserverid** | integer | Server ID of the database server where this data originated. This server ID is group ID from the sqlhosts file or the SQLHOSTS registry key. |
| **ctkeyid** | integer | Logical log ID |
| **ctkeypos** | integer | Position in the logical log on the source server for the transaction represented by the buffer |

(1 of 2)

| Column | Type | Description |
|---|---|---|
| **ctkeysequence** | integer | Sequence number for the buffer within the transaction |
| **ctstamp1** | integer | Together with ctstamp2, forms an insertion stamp that specifies the order of the transaction in the queue |
| **ctstamp2** | integer | Together with ctstamp1, forms an insertion stamp that specifies the order of the transaction in the queue |
| **ctcommittime** | integer | Time when the transaction represented by this buffer was committed |
| **ctuserid** | integer | Login ID of the user who committed this transaction |
| **ctfromid** | integer | Server ID of the server that sent this transaction. Used only in hierarchical replication. |

(2 of 2)

## Columns of the Buffer Tables

The tables whose names end with **_buf** give information about the buffers that form the transactions listed in the **_txn** table. All the **_buf** tables have the same columns and the same column definitions.

| Column | Type | Description |
|---|---|---|
| **cbflags** | integer | Internal flags for this buffer |
| **cbsize** | integer | Size of this buffer in bytes |
| **cbkeyserverid** | integer | Server ID of the database server where this data originated. This server ID is group ID from the sqlhosts file or the SQLHOSTS registry key. |
| **cbkeyid** | integer | Login ID of the user who originated the transaction represented by this buffer |
| **cbkeypos** | integer | Log position on the source server for the transaction represented by this buffer |

(1 of 2)

| Column | Type | Description |
|---|---|---|
| **cbkeysequence** | integer | Sequence number for this buffer within the transaction |
| **cbgroupid** | integer | Replicate or replicate group identifier for the data in this buffer |
| **cbcommittime** | integer | Time when the transaction represented by this buffer was committed |

(2 of 2)

The following columns combine to form the primary key for this table: **cbkeyserverid**, **cbkeyid**, **cbkeypos**, **cbkeysequence**.

The columns **cbkeyserverid**, **cbkeyid**, and **cbkeypos** form a foreign key that points to the transaction in the **_txn** table that the buffer represents.

# SMI Tables for Enterprise Replication

The following sections describe the individual tables of the **sysmaster** database that refer to Enterprise Replication.

## syscdrack_buf

The **syscdrack_buf** table contains information about the buffers that form the acknowledgment queue. When the target database server applies transactions, it sends an acknowledgment to the source database server. When the source database server receives the acknowledgment, it can then delete those transactions from its send queue.

For information on the columns of the **syscdrack_buf** table, refer to "Columns of the Buffer Tables" on page B-4.

## syscdrack_txn

The **syscdrack_txn** table contains information about the acknowledgment queue. When the target database server applies transactions, it sends an acknowledgment to the source database server. When the source database server receives the acknowledgment, it can then delete those transactions from its send queue.

The acknowledgment queue is an in-memory only queue. That is, it is a volatile queue that is lost if the database server is stopped.

For information on the columns of the **syscdrack_txn** table, refer to "Columns of the Transaction Tables" on page B-3.

## syscdrctrl_buf

The **syscdrctrl_buf** table contains buffers that provide information about the control queue. The control queue is a stable queue that contains control messages for the replication.

For information on the columns of the **syscdrctrl_buf** table, refer to "Columns of the Buffer Tables" on page B-4.

## syscdrctrl_txn

The **syscdrctrl_txn** table contains information about the control queue. The control queue is a stable queue that contains control messages for the replication.

For information on the columns of the **syscdrctrl_txn** table, refer to "Columns of the Transaction Tables" on page B-3.

## syscdrerror

The **syscdrerror** table contains information about errors that Enterprise Replication has encountered.

| Column | Type | Description |
|---|---|---|
| **errornum** | integer | Error number |
| **errorserv** | char(128) | Database server name where error occurred |
| **errorseqnum** | integer | Sequence number that can be used to prune single-error table |
| **errorstmnt** | char(128) | Error description |
| **error time** | datetime year to second | Time error occurred |
| **sendserv** | char(128) | Database server name, if applicable, that initiated error behavior |
| **reviewed** | char(1) | Y if reviewed and set by DBA<br>N if not reviewed |

## syscdrgrp

The **syscdrgrp** table contains replicate group information.

| Column | Type | Description |
|---|---|---|
| **grpname** | char(218) | Replicate group name |
| **isseq** | char(1) | Y if sequential<br>N if non-sequential |

## syscdrgrppart

The **syscdrgrppart** table contains replicate group information.

| Column | Type | Description |
|---|---|---|
| **grpname** | char(128) | Replicate group name |
| **servername** | char(128) | Database server name |
| **state** | integer | The following state values apply when the OR operator is applied:<br>0x0000 = Defined in the catalog<br>0x0001 = Peer request failed; definition persists<br>0x0002 = Inactive state<br>0x0004 = Active state<br>0x0008 = Database server is suspended<br>0x00010 = Marked for deletion and waiting for cleanup to occur<br>0x00020 = Temporary state (in transition) |

## syscdrpart

The **syscdrspart** table contains participant information.

| Column | Type | Description |
|---|---|---|
| **replname** | char(128) | Replicate name |
| **servername** | char(128) | Database server name |
| **partstate** | char(18) | Participant state: ACTIVE, INACTIVE |
| **partmode** | char(1) | P = primary database server (read / write)<br>R = target database server (read only) |
| **usetabowner** | integer | Use table owner to derive permissions<br>1 = enabled<br>0 = not enabled |
| **dbsname** | char(128) | Database name |

(1 of 2)

| Column | Type | Description |
|---|---|---|
| **owner** | char(32) | Owner name |
| **tabname** | char(128) | Table name |
| **selectstmnt** | text | SELECT statement for this participant |

(2 of 2)

## syscdrprog

The **syscdrprog** table lists the contents of the Enterprise Replication progress tables. The progress tables keep track of what data has been sent to which servers and which servers have acknowledged receipt of what data. Enterprise Replication uses the transaction keys and stamps to keep track of this information.

The progress table is two-dimensional. For each server to which Enterprise Replication sends data, the progress tables keep progress information on a per-replicate basis.

| Column | Type | Description |
|---|---|---|
| **dest_id** | integer | Server ID of the destination server |
| **group_id** | integer | The ID that Enterprise Replication uses to identify the group for which this information is valid |
| **source_id** | integer | Server ID of the server from which the data originated |
| **key_acked_srv** | integer | Last key for this group that was acknowledged by this destination |
| **key_acked_lgid** | integer | Logical log ID |
| **key_acked_lgpos** | integer | Logical log position |

(1 of 2)

| Column | Type | Description |
|---|---|---|
| **key_acked_seq** | integer | Logical log sequence |
| **tx_stamp_1** | integer | Together with tx_stamp2, forms the stamp of the last transaction acknowledged for this group by this destination |
| **tx_stamp_2** | integer | Together with tx_stamp1, forms the stamp of the last transaction acknowledged for this group by this destination |

(2 of 2)

## syscdrq

The **syscdrq** table contains information about Enterprise Replication queues.

| Column | Type | Description |
|---|---|---|
| **srvid** | integer | The identifier number of the database server |
| **repid** | integer | The identifier number of the replicate |
| **srcid** | integer | The server ID of the source database server<br>In cases where a particular server is forwarding data to another server, srvid is the target and srcid is the source that started the transaction. |
| **srvname** | char(128) | The name of the database server |
| **replname** | char(128) | Replicate group or replicate name |
| **srcname** | char(128) | The name of the source database server |

## syscdrqueued

The **syscdrqueued** table contains data-queued information.

| Column | Type | Description |
|---|---|---|
| **servername** | char(128) | Sending to database server name |
| **name** | char(128) | Replicate group or replicate name |
| **bytesqueued** | decimal(32,0) | Number of bytes queued for the server servername |

## syscdrrecv_buf

The **syscdrrecv_buf** table contains buffers that provide information about the data-receive queue. When a replication server receives replicated data from a source database server, it puts this data on the receive queue for processing. On the target side, Enterprise Replication picks up transactions from this queue and applies them on the target.

For information on the columns of the **syscdrrecv_buf** table, refer to "Columns of the Buffer Tables" on page B-4.

## syscdrrecv_txn

The **syscdrrecv_txn** table contains information about the data receive queue. The receive queue is an in-memory queue.

When a replication server receives replicated data from a source database server, it puts this data on the receive queue, picks up transactions from this queue, and applies them on the target.

For information on the columns of the **syscdrrecv_txn** table, refer to "Columns of the Transaction Tables" on page B-3.

## syscdrrepl

The **syscdrrepl** table contains replicate information.

| Column | Type | Description |
|---|---|---|
| **replname** | char(18) | Replicate name |
| **grpname** | char(128) | Replicate group name (if replicate is part of a replicate group)<br>Null if replicate is not part of a replicate group |
| **replstate** | char(18) | Replicate state. For possible values, refer to "cdr list replicate" on page 11-48. |
| **freqtype** | char(1) | Type of replication frequency:<br>C = continuous<br>I = interval<br>T = time based<br>M = day of month<br>W = day of week |
| **freqmin** | integer | Minute (after the hour) refresh should occur<br>Null if continuous |
| **freqhour** | integer | Hour refresh should occur<br>Null if continuous |
| **freqday** | integer | Day of week or month refresh should occur |
| **scope** | char(1) | Replication scope:<br>T = transaction<br>R = row-by-row |
| **invokerow-spool** | char(1) | Y = row spooling is invoked<br>N = no row spooling is invoked |
| **invoke transpool** | char(1) | Y = transaction spooling is invoked<br>N = no transaction spooling is invoked |
| **primresolution** | char(1) | Type of primary conflict resolution:<br>I = ignore<br>T = time stamp<br>S = stored procedure |

(1 of 2)

| Column | Type | Description |
|---|---|---|
| **secresolution** | char(1) | Type of secondary conflict resolution:<br>S = stored procedure<br>Null = not configured |
| **storedprocname** | char(128) | Name of stored procedure for secondary conflict resolution<br>Null if not defined. |
| **istriggerfire** | char(1) | Y = triggers are invoked<br>N = triggers are not invoked |
| **iscanonical** | char(1) | Y = conversion to canonical form is required<br>N = no canonical form used |

(2 of 2)

## syscdrs

The **syscdrs** table contains information about database servers that have been declared to Enterprise Replication.

| Column | Type | Description |
|---|---|---|
| **servid** | integer | Server identifier |
| **servname** | char(128) | Database server name |
| **cnnstate** | char(1) | Status of connection to this database server:<br>C = Connected<br>D = Connection disconnected (will be retried)<br>T = Idle time-out caused connection to terminate<br>X = Connection closed by user command<br>Connection unavailable until reset by user |
| **cnnstatechg** | integer | Time that connection state was last changed |
| **servstate** | char(1) | Status of database server:<br>A = Active<br>S = Suspended<br>H = Holding<br>Q = Quiescent (initial sync state only) |

(1 of 2)

| Column | Type | Description |
|---|---|---|
| **ishub** | char(1) | Y = the server is a hub<br>N = the server is not a hub<br>A hub is any replication server that forwards information to another replication server. |
| **isleaf** | char(1) | Y = the server is a leaf server<br>N = the server is not a leaf server |
| **rootserverid** | integer | The identifier of the root server |
| **forwardnodeid** | integer | The identifier of the parent server |
| **timeout** | integer | Idle timeout |

(2 of 2)

Although not directly connected, a nonroot server is similar to a root server except it forwards all replicated messages through its parent (root) server. All nonroot servers are known to all root servers and other nonroot servers. A nonroot server can be a terminal point in a tree or it can be the parent for another nonroot server or a leaf server. Nonroot and root servers are aware of all replication servers in the replication environment, including all the leaf servers.

A leaf server is a nonroot server that has a sparse catalog. A leaf server has knowledge only of itself and its parent server. It does not contain information about replicates of which it is not a participant. The leaf server must be a terminal point in a replication hierarchy.

## syscdrsend_buf

The **syscdrsend_buf** table contains buffers that give information about the send queue. When a user performs transactions on the source database server, Enterprise Replication queues the data on the send queue for delivery to the target servers. The send queue is a *semi-stable queue*. A semi-stable queue is a queue that is in memory until CDR_QUEUEMEM is reached and spooled thereafter.

For information on the columns of the **syscdrsend_buf** table, refer to "Columns of the Buffer Tables" on page B-4.

## syscdrsend_txn

The **syscdrsend_txn** table contains information about the send queue. When a user performs transactions on the source database server, Enterprise Replication queues the data on the send queue for delivery to the target servers. The send queue is a semi-stable queue.

For information on the columns of the **syscdrsync_txn** table, refer to "Columns of the Transaction Tables" on page B-3.

## syscdrserver

The **syscdrserver** table contains information about database servers that have been declared to Enterprise Replication.

| Column | Type | Description |
|---|---|---|
| **servid** | integer | Replication server ID |
| **servername** | char(128) | Database server group name |
| **connstate** | char(1) | Status of connection to this database server:<br>C = Connected<br>D = Connection disconnected (will be retried)<br>T = Idle time-out caused connection to terminate<br>X = Connection closed by user command<br>Connection unavailable until reset by user |
| **connstate-change** | integer | Time that connection state was last changed |
| **servstate** | char(1) | Status of this database server:<br>A = Active<br>S = Suspended<br>H = Holding<br>Q = Quiescent (initial sync state only) |
| **ishub** | char(1) | Y = server is a hub<br>N = server is not a hub |
| **isleaf** | char(1) | Y = the server is a leaf server<br>N = the server connection is not a leaf server |

(1 of 2)

| Column | Type | Description |
|---|---|---|
| **rootserverid** | integer | The identifier of the root server |
| **forwardnodeid** | integer | The identifier of the parent server |
| **idletimeout** | integer | Idle time out |
| **sendqueueloc** | char(128) | Dbspace name that indicates where the send queue is located |
| **rcvqueueloc** | char(128) | Dbspace name that indicates where the receive queue is located |
| **atsdir** | char(256) | ATS directory spooling name |
| **risdir** | char(256) | RIS directory spooling name |

(2 of 2)

## syscdrsync_buf

The **syscdrsync_buf** table contains buffers that give information about the synchronization queue. Enterprise Replication uses this queue only when defining a replication server and synchronizing it with another replication server.

For information on the columns of the **syscdrsync_buf** table, refer to "Columns of the Buffer Tables" on page B-4.

## syscdrsync_txn

The **syscdrprog** table contains information about the synchronization queue. This queue is currently used only when defining a replication server and synchronizing it with another replication server. The synchronization queue is an in-memory-only queue.

For information on the columns of the **syscdrsync_txn** table, refer to "Columns of the Transaction Tables" on page B-3.

## syscdrtx

The **syscdrtx** table contains information about Enterprise Replication transactions.

| Column | Type | Description |
|---|---|---|
| **srvid** | integer | Server ID |
| **srvname** | char(128) | Name of database server from which data is received |
| **txprocssd** | integer | Transaction processed from database server *srvname* |
| **txcmmtd** | integer | Transaction committed from database server *srvname* |
| **txabrtd** | integer | Transaction aborted from database server *srvname* |
| **rowscmmtd** | integer | Rows committed from database server *srvname* |
| **rowsabrtd** | integer | Rows aborted from database server *srvname* |
| **txbadcnt** | integer | Number of transactions with source commit time (on database server *srvname*) greater than target commit time |

## syscdrtxproc

The **syscdrtxproc** table contains information about an Enterprise Replication transaction process.

| Column | Type | Description |
|---|---|---|
| **servername** | char(128) | Receiving from database server name |
| **txprocessed** | integer | Transaction processed from database server *servername* |
| **txcommitted** | integer | Transaction committed from database server *servername* |
| **txaborted** | integer | Transaction aborted from database server *servername* |

(1 of 2)

| Column | Type | Description |
|---|---|---|
| **rowscommitted** | integer | Rows committed from database server *servername* |
| **rowsaborted** | integer | Rows aborted from database server *servername* |
| **txbadcnt** | integer | Number of transactions with source commit time (on database server *servername*) greater than target commit time |

(2 of 2)

Appendix

C

# Return Codes

## Definition of Return Codes

The "List of Return Values" on page C-2 shows the return messages and codes that Enterprise Replication returns. These messages are also documented in the **cdrerr.h** file in **$INFORMIXDIR/incl/esql** or **%INFORMIXDIR%\incl\esql** directory.

### Using the Stored Error Information

The following API functions include a pointer (called *which*) that points to a structure that contains error information when information is available. The **struct** column in the following table indicates the structure where error information will be stored.

- **cdr_define_group**()
- **cdr_define_repl**()
- **cdr_modify_replmode**()
- **cdr_participate_group**()
- **cdr_participate_repl**()
- **cdr_resume_serv**()
- **cdr_start_group**()
- **cdr_start_repl**()
- **cdr_stop_group**()
- **cdr_stop_repl**()
- **cdr_suspend_serv**()

## List of Return Values

| Mnemonic | Numeric Value | Description | struct |
|---|---|---|---|
| CDR_SUCCESS | 0 | Command was successful | |
| CDR_ENOCONNECT | 1 | A connection does not yet exist for the given server | |
| | 2 | (not in use) | |
| CDR_ECOLUNDEF | 3 | Table column undefined | |
| CDR_ECOMPAT | 4 | Incompatible server version | |
| CDR_ECONNECT | 5 | Unable to connect to server specified | |
| CDR_EDBDNE | 6 | Database does not exist | **CDR_plist** |
| CDR_EDBLOG | 7 | Database not logged | **CDR_plist** |
| CDR_EFREQ | 8 | Bad or mismatched frequency attributes<br>The frequency type specified is incorrect or the values are out of range for the frequency type specified. | **CDR_idlist** |
| CDR_ECONNECTED | 9 | A connection already exists for the given server | |
| CDR_EGRPSTATE | 10 | Illegal group state change | |
| CDR_EGRPUNDEF | 11 | Undefined group | |
| CDR_EGRPUNIQ | 12 | Group name already in use | |
| CDR_EIDLE | 13 | Invalid idle time specification | |
| CDR_EINVOP | 14 | Invalid operator or specifier | |
| | 15 | (not in use) | |
| | 16 | (not in use) | |
| CDR_ENOPART | 17 | Participants required for operation specified | |
| CDR_ENOPKEY | 18 | Table does not contain primary key | **CDR_plist** |

(1 of 6)

| Mnemonic | Numeric Value | Description | struct |
|---|---|---|---|
| CDR_EOWNER | 19 | Table does not exist<br>No namespace in database for owner specified. | **CDR_plist** |
| CDR_EPACTIVE | 20 | Server already participating in replicate | **CDR_plist** |
| | 21 | (not in use) | |
| CDR_EPKEYSELCT | 22 | Primary key not contained in select clause | |
| | 23 | (not in use) | |
| | 24 | (not in use) | |
| CDR_EREPACTIVE | 25 | Replicate already participating in a group | **CDR_idlist** |
| CDR_EREPDEF | 26 | Group operation not permitted on replicate | **CDR_idlist** |
| | 27 | (not in use) | |
| CDR_EREPLUNIQ | 28 | Replicate name already in use | |
| CDR_ETBLDNE | 29 | Table does not exist | **CDR_plist** |
| CDR_EREPSTATE | 30 | Illegal replicate state change | **CDR_idlist** |
| CDR_EREPUNDEF | 31 | Undefined replicate | **CDR_idlist** |
| CDR_ESENDQ | 32 | dbspace specified for the send queue does not exist | |
| CDR_ESERVDEF | 33 | Server not participant in replicate / group | **CDR_idlist** |
| | 34 | (not in use) | |
| CDR_ESERVRESOLV | 35 | Server not defined in sqlhosts | |
| CDR_ESERVSET | 36 | Disjoint servers for replicates | **CDR_idlist** |
| CDR_ESERVUNDEF | 37 | Undefined server | **CDR_plist** |
| CDR_ESPDNE | 38 | Stored procedure does not exist<br>Undefined stored procedure. | **CDR_conf** |

| Mnemonic | Numeric Value | Description | struct |
|---|---|---|---|
| CDR_ESQLSYN | 39 | Illegal select syntax<br>The SELECT statement uses illegal syntax. | **CDR_plist** |
| CDR_ESQLUNSUP | 40 | Unsupported SQL syntax (join, and so on) | |
| | 41 | (not in use) | |
| CDR_ETIME | 42 | Invalid time | |
| CDR_EVALID | 43 | Participants required for specified operation<br>The **R_PARTICIPANT** flag bit is set in **mflags** but the **participants** argument is null. | |
| CDR_ENAMERR | 44 | Illegal name syntax<br>Invalid replicate or database server name. | **CDR_plist** |
| CDR_EPART | 45 | Invalid participant | |
| | 46 | (not in use) | |
| CDR_ESERV | 47 | Invalid server name | |
| CDR_ENOMEM | 48 | Out of memory | |
| CDR_EREPMAX | 49 | Maximum number of replicates exceeded | |
| CDR_EPARTMAX | 50 | Maximum participants | |
| CDR_ERUMOR | 51 | Attempt to delete server remotely | |
| CDR_ESERVUNIQ | 52 | Server name already in use | |
| CDR_EDUPL | 53 | Duplicate server or replicate | |
| CDR_EBADCRULE | 54 | Bad conflict rule specified | |
| CDR_ENOSCOPE | 55 | Resolution scope not specified | |
| CDR_ESCOLSDNE | 56 | Shadow columns do not exist for table<br>Shadow columns are not present or have invalid data types. If non-ignore conflict resolution is chosen, shadow columns must exist for replicated table. | |

| Mnemonic | Numeric Value | Description | struct |
|---|---|---|---|
| CDR_ECRDELTAB | 57 | Error creating delete table | |
| CDR_ENOCRULE | 58 | No conflict resolution rule specified | |
| CDR_EBADSCOLS | 59 | Table has bad type for shadow columns or has shadow columns at wrong place | |
| CDR_EGRPPART | 60 | Illegal operation on group participant | |
| CDR_ENOPERM | 61 | User does not have permission to issue command | |
| CDR_ENOCDR | 62 | Enterprise Replication not active<br>Make sure that the **INFORMIXSERVER** environment variable is set to a database server that is:<br>■ a member of a valid database server group;<br>■ the DBSERVERNAME in the configuration file;<br>■ not a shared-memory connection | |
| CDR_ECDR | 63 | Enterprise Replication already active<br>The database server that you tried to define is already active in Enterprise Replication. | |
| CDR_ENOSYNC | 64 | Remote / cyclic synchronization not allowed | |
| CDR_ESERVID | 65 | Server identifier already in use | |
| CDR_ENOTIME | 66 | No upper time for prune error | |
| CDR_ERRNOTFOUND | 67 | Error not found for delete or update | |
| CDR_EPARTMODE | 68 | Illegal participant mode | |
| CDR_ECONFLICT | 69 | Conflict mode for replicate not ignore | |
| CDR_ECONSAME | 70 | Connect / disconnect to / from same server | |
| CDR_EROOT | 71 | Conflicting root server flags | |
| CDR_EPARENT | 72 | Cannot delete server with children | |

(4 of 6)

| Mnemonic | Numeric Value | Description | struct |
|---|---|---|---|
| CDR_ESPAROOT | 73 | Leaf-root configuration not allowed | |
| | 74 | (not in use) | |
| CDR_ELIMITED | 75 | Request denied on limited server | |
| CDR_EMSGFORMAT | 76 | Unsupported message format | |
| CDR_EDROPDB | 77 | Could not drop **syscdr** database | |
| CDR_EATSDIR | 78 | ATS directory does not exist | |
| CDR_ERISDIR | 79 | RIS directory does not exist | |
| CDR_ECRCHANGE | 80 | Illegal conflict resolution change | |
| CDR_EUDTBADCOL | 81 | UDT collection types not allowed | |
| CDR_EUDTEXPR | 82 | UDTs not allowed in expressions (such as where clauses) | |
| CDR_ENOUDTPKEY | 83 | No UDTs in primary key allowed<br>The primary key for a replicated table cannot contain UDTs. | |
| CDR_ESYNC | 84 | No sync server with nonroot and leaf | |
| CDR_EPARTFLAGS | 85 | Incorrect participant flags | |
| CDR_ELEAF | 86 | Conflicting leaf server flags or attempt to sync with leaf server | |
| CDR_ESTOPFLAGS | 87 | Invalid cdr stop options | |
| | 88 | (not in use) | |
| CDR_EMODRECVQ | 89 | Cannot modify dbspace for queues | |
| CDR_ECLOCKSKEW | 90 | System clocks are out of synchronization | |
| CDR_ESERVSTATE | 91 | Invalid server state transition | |
| CDR_ESERVERR | 100 | Fatal server error | |

| Mnemonic | Numeric Value | Description | struct |
|---|---|---|---|
| CDR_ENOSUPPORT | 101 | Unsupported feature | |
| CDR_EINVSYNC | 102 | Root server cannot sync with nonroot or leaf servers | |
| CDR_EINVCONNECT | 103 | Invalid server to connect | |
| CDR_ETEMPDB | 104 | Cannot use temp dbspaces for Send/Recv Queues<br>This error also occurs if you try to move the Error Table into a temporary dbspace. | |

(6 of 6)

Appendix

D

# Fine-Tuning Enterprise Replication

This appendix provides reference material and several examples to help you fine-tune Enterprise Replication.

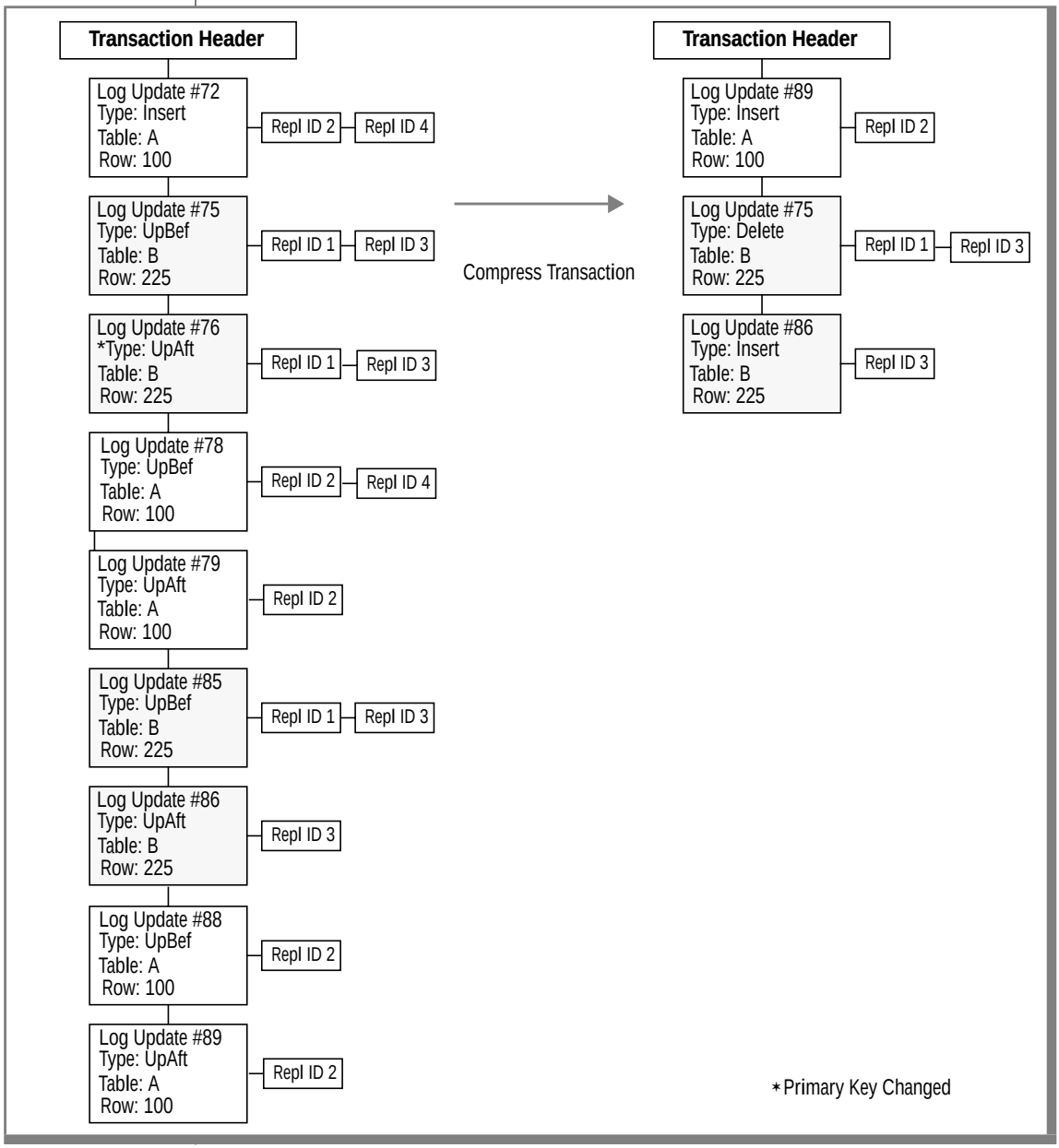
## Evaluating Images in a Complex Transaction

Figure D-1 on page D-2 illustrates the evaluation compression of a single transaction with the following activities:

- One row is inserted and then updated several times.
- Another row in a different table is updated twice and the primary key changes in the second update.

Each row is evaluated against the appropriate replicates, and a list of replicates is assembled where the row has been evaluated as a yes.

***Figure D-1***
*How Enterprise Replication Evaluates Images in a Complex Transaction*



# A Stored-Procedure Example

This example illustrates the function of conflict resolution by stored procedure. The stored procedure will resolve a conflict for rows in a table defined as:

```
CREATE TABLE employees
        (
        emp_num int,
        empname CHAR(15),
        salary INT,
        PRIMARY KEY (emp_num)
       ) WITH CRCOLS LOCK MODE ROW;
```

The replicate is defined as `SELECT * FROM employees`. The `emp_num` is the primary key. The primary key is passed with the other replicated row data and the local target row data.

The procedure determines the type of collision. If a replication order error is detected, the replicated row is not applied but is logged. Otherwise, if the replicated salary field is not equal to the local target, the replicated row is applied but is not logged.

```
create procedure updateSalary
(localServer CHAR(18), localTS DATETIME YEAR TO SECOND,
localDelete CHAR(1),
 replServer CHAR(18), replTS DATETIME YEAR TO SECOND, replDbop
CHAR(1),
 localEmpNum INT, localEmpName CHAR(15), localSalary INT,
 replEmpNum  INT, replEmpName  CHAR(15), replSalary  INT)
returning CHAR(1), INT, INT, CHAR(15), INT;

{-------------------------------------------------------}
{ Do nothing if replicated row is out of logical order:
     (1) if local row is found in the "deleted" row table
                 and
                 the replicated operation is update or delete
     (2) if local row is found in target table and the
                 replicated operation is insert.
        Enter the row to log.
}

if (localDelete = 'Y') then
    if (replDbop = 'U') or (replDbop ='D')
        then
            return 'O', 100, null, null, null;
    end if
if (replDbop = 'I') then
    return 'O', 101, null, null, null;
end if

{-------------------------------------------------------}
{ If the replicated row is from server "corporate" and the
  the local row's salary field is not equal to the
    replicated salary field, then update local row.
}

if (replServer = 'corporate' and localSalary != replSalary)
then
        return replDbop, 200, replEmpNum, replEmpName,
replSalary;
else
        return replDbop, 201, replEmpNum, replEmpName,
            localSalary;
end if;

end procedure
```

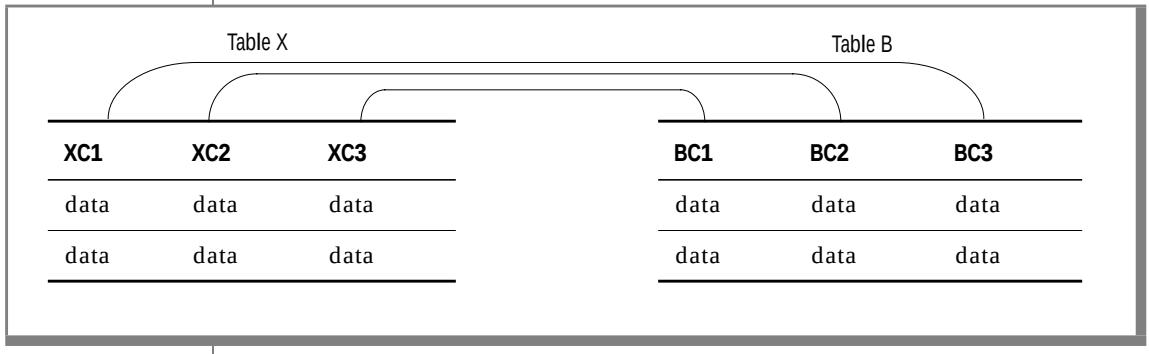
# Column Mapping

You can reorder columns between replicated tables. Each participant in the replication system must define the identical number of columns (n to n mappings only). These columns must define the same domains. No data conversions are performed by column mapping.

Column mapping is defined by the enumerations of columns in the select statement that accompanies each participant in a replication system.

For example in Figure D-2, **Table X** with columns **XC1**, **XC2**, and **XC3** maps to columns **BC1**, **BC2**, and **BC3** in **Table B**.

***Figure D-2***
*Column Mapping*




The columns map as follows: **XC1** to **BC3**, **XC2** to **BC2**, and **XC3** to **BC1**. The SELECT statements would then read:

```
SELECT XC1, XC2, XC3 FROM Table X
SELECT BC3, BC2, BC1 FROM Table B.
```

# Step-by-Step Procedure

Replication is defined and controlled with a few simple steps. The following example shows three sites in the replication system:

- Bangor, Maine
- Ames, Iowa
- Milpitas, California

Each site has a database named **general**. Each site has equivalent **sqlhosts** files (UNIX) or SQLHOSTS registry key entries (Windows NT) that identify each of the database servers and server groups. The table **employees** is to be replicated among all three sites.

The database **general** has the following schema:

```
CREATE DATABASE general WITH LOG; {or OPEN database general,
if already created}
CREATE TABLE employees
        (
        emp_num     INT,
        empname     CHAR(15),
        company     CHAR(20),
        address1    CHAR(20),
        address2    CHAR(20),
        city        CHAR(15),
        state       CHAR(2),
        zipcode     CHAR(5),
        phone       CHAR(18),
        PRIMARY KEY (emp_num)
      ) WITH CRCOLS LOCK MODE ROW;
```

## Step 1: Define Database Schema for Maine Site

On the database server in Bangor, Maine, create database **general**.

## Step 2: Define Database Schema for Iowa Site

On the database server in Ames, Iowa, create database **general**.

## Step 3: Define Database Schema for California Site

On the database server in Milpitas, California, create database **general**.

## Step 4: Declare Database Servers to Enterprise Replication

Declare all three database servers (**bangor**, **ames**, and **milpitas**) to Enterprise Replication. This step is required because none of the database servers has been previously declared. You can declare the database servers using any of the following methods:

- Using the Enterprise Replication Manager
- Using the CLU command **cdr define server**
- Using the API call **cdr_define_serv()**

After the database servers are declared to Enterprise Replication, each database server has a database named syscdr that contains the global catalog.

## Step 5: Define a Replicate Called employeeinfo

Next the replication system is defined using the API call **cdr_define_repl()**. The replication system will be defined with the name **employeeinfo** with participant database servers **bangor**, **ames**, and **milpitas** as follows:

```
general@bangor:employees "SELECT * FROM employees"
general@milpitas:employees "SELECT * FROM employees"
general@ames:employees "SELECT * FROM employees"
with resolution rules set to timestamp primary
```

The replication system is defined in the global catalog with an initial state of *inactive*. This state means that no replication data is captured or transmitted.

## Step 6: Prepare the Tables for Replication

The database administrator must ensure that the data in the tables is consistent across all sites before starting replication.

This example assumes that **bangor** contains the complete correct version of the table. The following list shows actions necessary to prepare for replication:

- Lock the tables on **bangor**, **ames**, **milpitas** with a shared lock.
- Unload the authoritative copy of the table from **bangor**.
- Load new copies into **ames** and **milpitas**.

## Step 7: Start Replication

The API call **cdr_start_repl()** is invoked with the name of the replicate. This command starts data capture and queueing at all participating database servers.

Unlock the tables on **bangor**, **ames**, and **milpitas.**

After the completion of the start, the replication system is active. Data is being captured and propagated.

## Step 8: Suspend or Remove Database Servers

Some months later the database administrator is informed that the computer bangor must be taken out of service for maintenance. The administrator has several choices:

- Suspend the bangor database server for both milpitas and ames.
- Remove bangor as a participant from employeeinfo.
- Delete the database server bangor from the global catalog.

### Method 1 Uses the cdr_suspend_serv() Call

If the database administrator monitors the queues on bangor after the suspends to ames and milpitas are issued and ensures that the queues have emptied properly, no data intended for **bangor** is lost. All updates for **bangor** are queued at the respective origins (either **ames** or **milpitas**). When **bangor** returns to operation and is resumed at **ames** and **milpitas**, all updates that occurred during the **bangor** outage are propagated to **bangor**. No resynchronization is necessary. The database server is suspended on both of the other database servers (**ames** and **milpitas**). The replication is still in the active state.

### Method 2 or 3 Assumes bangor Is Resynchronized

These methods assume that **bangor** is resynchronized before it is re-added to the **employeeinfo** replicate on **ames** and **milpitas**. Method 2 uses **cdr_participate_repl()** to remove **bangor** as a participant in the **employeeinfo** replicate. Method 3 uses **cdr_delete_serv()** to remove **bangor** as a participant in any replication. In both cases replication messages intended for **bangor** might be discarded. The replication is still active on **ames** and **milpitas**.

## Step 9: Stop Replication with cdr_stop_repl()

When the replication is no longer required, the administrator can issue the **cdr_stop_repl()** call. This API stops data capture at **bangor**, **ames**, and **milpitas**. After all table changes have propagated (queues have emptied), the tables will be consistent. During the period that the queues are emptying, the replication is quiescent; no data is being captured but data is still being propagated for the replication. When the queues have drained on all the sites, the replication returns to the inactive state.

WIN NT

# SQLHOSTS Registry Key

When you install the database server, the **setup** program creates the following key in the Windows NT registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\INFORMIX\SQLHOSTS
```

This branch of the HKEY_LOCAL_MACHINE subtree stores the **sqlhosts** information. Each key on the SQLHOSTS branch is the name of a database server. When you click the database server name, the registry displays the values of the HOST, OPTIONS, PROTOCOL, and SERVICE fields for that particular database server.

Each computer that hosts a database server or a client must include the connectivity information either in the SQLHOSTS registry key or in a central registry. When the client application runs on the same computer as the database server, they share a single SQLHOSTS registry key.

## The Location of the SQLHOSTS Registry Key

When you install the database server on Windows NT, the installation program asks where you want to store the SQLHOSTS registry key. You can specify one of the following two options:

- The local computer where you are installing the database server
- Another computer in the network that serves as a central, shared repository of **sqlhosts** information for multiple database servers in the network

Using a shared SQLHOSTS registry key relieves you of the necessity to maintain the same **sqlhosts** information on multiple computers. However, the **hosts** and **services** files on *each* computer must contain information about all computers that have database servers.

If you specify a shared SQLHOSTS registry key, you must set the **INFORMIX-SQLHOSTS** environment variable on your local computer to the name of the Windows NT computer that stores the registry. The database server first looks for the SQLHOSTS registry key on the INFORMIXSQLHOSTS computer. If the database server does not find an SQLHOSTS registry key on the INFORMIX-SQLHOSTS computer, or if **INFORMIXSQLHOSTS** is not set, the database server looks for an SQLHOSTS registry key on the local computer.

You must comply with Windows NT network-access conventions and file permissions to ensure that the local computer has access to the shared SQLHOSTS registry key. For information about network-access conventions and file permissions, see your Windows NT documentation.

## Preparing the SQLHOSTS Connectivity Information

With Dynamic Server, Version 7.31, use the IECC Add Database Servers wizard to prepare the SQLHOSTS connectivity information.

With Dynamic Server, Version 9.2, refer to the documentation notes described in "Documentation Notes, Release Notes, Machine Notes" on page 15 of the Introduction for information about preparing the SQLHOSTS connectivity information.♦

Alternatively, you can use the Windows NT program **regedt32**.

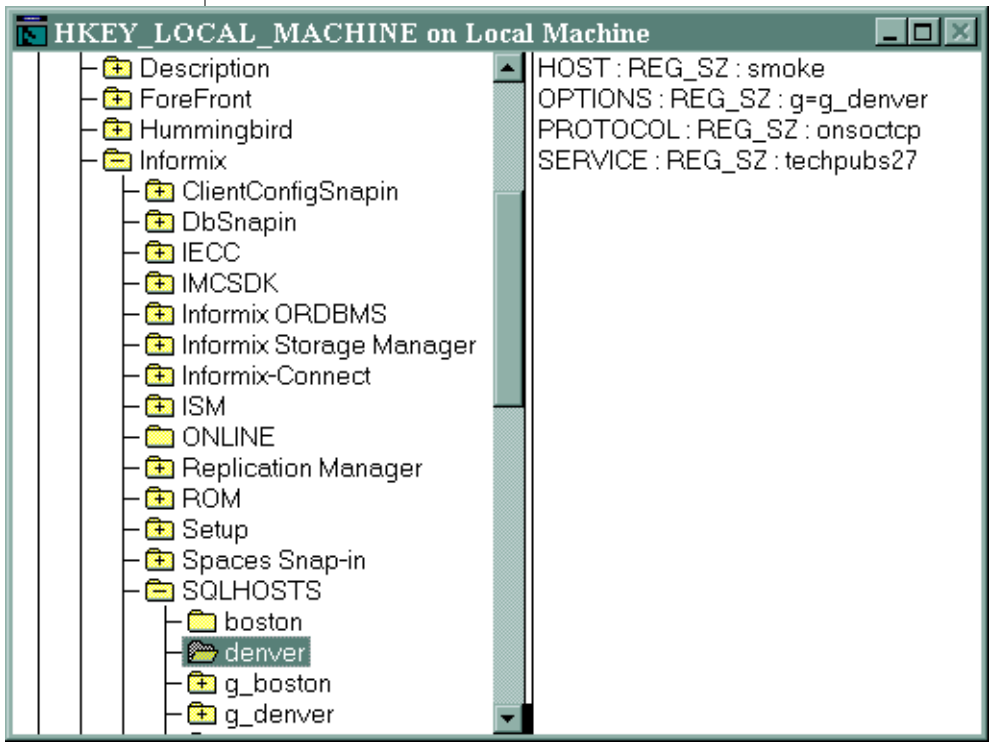
***Important:*** *Informix strongly recommends that you use one of the tools that Informix provides to prepare the SQLHOSTS connectivity information. Unless you use extreme caution,* ***regedt32*** *can destroy the configuration, not only of your Informix products, but of your other applications.*

### To add connectivity information using regedt32

1. Run **regedt32**.
2. In the Registry Editor window, select the window for the HKEY_LOCAL_MACHINE subtree.
3. Click the folder icons to select the \SOFTWARE\INFORMIX\SQLHOSTS branch.
4. With the SQLHOSTS key selected, choose **Edit→Add Key**.
5. In the Add Key dialog box, enter the name of the database server in the Key Name dialog box. Leave the Class dialog box blank. Click **OK**.
6. Select the new key that you just made (the key with the database server name).
7. Choose **Edit→Add Value.**
8. In the Add Value dialog box, enter one of the fields of the **sqlhosts** information (HOST, OPTIONS, PROTOCOL, SERVICE) in the Value Name dialog box. Do not change the Data Type box. Click **OK**.
9. In the String Editor dialog box, type the value for the field that you selected and click **OK**.
10. Repeat steps 8 and 9 for each field of the **sqlhosts** information.

Figure E-1 illustrates the location and contents of the SQLHOSTS registry key for the database server **payroll**.

***Figure E-1***
*sqlhosts Information in the Windows NT Registry*




After you create the registry key for the database server, you must make a database server group that includes the database server. For more information, refer to "Database Server Groups" on page 11-12.

1. With the SQLHOSTS key selected, choose **Edit→Add Key**.
2. In the Add Key dialog box, enter the name of the database server group in the Key Name dialog box. In this manual (and in Figure E-1 on page E-4), each of the names of the database server groups are the database server names prefixed by **g_**. The **g_** prefix is not a requirement; it is just the convention that this manual uses.

   Leave the Class dialog box blank. Click **OK**.

3. Select the new key that you just made (the key with the database server group name).
4. Choose **Edit→Add Value.**
5. In the Add Value dialog box, enter one of the fields of the **sqlhosts** information (HOST, OPTIONS, PROTOCOL, SERVICE) in the Value Name dialog box. Do not change the Data Type dialog box. Click **OK**.
6. In the String Editor dialog box, type the value for the field that you selected and click **OK**.

   For a database server group, enter the following values:

   ```
   HOST        -
   OPTIONS     i=unique-integer-value
   PROTOCOL    group
   SERVICE     -
   ```

   Each database server group must have an associated identifier value (i=) that is unique among all database servers in your environment. Enter a minus (-) for HOST and SERVICE to indicate that you are not assigning specific values to those fields.
7. Repeat steps 6 and 7 for each field of the **sqlhosts** information.
8. Exit from the Registry Editor.

Appendix

F

# Enterprise Replication onstat -g Options

This appendix describes forms of the **onstat** command that are relevant to Enterprise Replication. The **onstat** utility reads shared-memory structures and provides statistics about the database server that are accurate at the instant that the command executes. The *system-monitoring interface* (SMI) also provides information about the database server. For general information about **onstat** and SMI, refer to the *Administrator's Reference*.

## The onstat -z Command

The **onstat -z** command clears database server statistics, including statistics that relate to Enterprise Replication.

## The onstat -g Command

The **onstat -g** commands are provided for support and debugging only. You can include only one of these options with each **onstat -g** command. This appendix describes the following **onstat -g** commands:

- **onstat -g cat**
- **onstat -g ddr**
- **onstat -g dss**
- **onstat -g dtc**
- **onstat -g grp**

- **onstat -g nif**
- **onstat -g que**
- **onstat -g rcv**
- **onstat -g rep**
- **onstat -g rqm**

## The onstat -g cat Command

The -g cat option of the onstat command prints a summary of information about servers, replicates, replicate groups and their characteristics.

The onstat -g cat command has the following formats:

```
onstat -g cat
onstat -g cat scope
onstat -g cat replname
```

The following table describes replname and scope.

| Modifier | Description |
|---|---|
| *replname* | The name of a replicate |
| *scope* | One of the following values:<br>servers — Print information on servers only<br>repls — Print information on replicates only<br>full — Print expanded information for both replicate servers and replicates |

## The onstat -g ddr Command

The **-g ddr** option of the **onstat** command is an important status tool. It prints the status of the Enterprise Replication database log reader. (The *ddr* refers to an internal component of Enterprise Replication that reads the log.) If log reading is blocked, then data might not be replicated until the problem is resolved. If the block is not resolved, then the database server might overwrite the read position of Enterprise Replication, which means that data will not be replicated. Once this occurs, you must manually resynchronize the source and target databases.

## The onstat -g dss Command

The **-g dss** option of the **onstat** command prints detailed information about the activity of individual data sync threads.

## The onstat -g dtc Command

The **-g dtc** option of the **onstat** command prints statistics about the delete table cleaner. The **-g dtc** option is used primarily as a debugging tool and by Informix Technical Support.

## The onstat -g grp Command

The **-g grp** option of the **onstat** command prints statistics about the grouper for debugging and technical support use.

The **onstat -g grp** command has the following formats:

```
onstat -g grp
onstat -g grp modifier
```

The modifier can have the values that the following table shows.

| Modifier | What the Command Prints |
|---|---|
| A | All the information printed by the G, T, P, E, R, and S modifiers |
| E | Grouper evaluator statistics |
| G | Grouper general statistics |
| L | Grouper global list |
| Lx | Grouper global list - expand open transactions |
| M | Grouper compression statistics |
| Mz | Clear grouper compression statistics |
| P | Grouper table partition statistics |
| R | Grouper replicate statistics |
| S | Grouper serial list head |
| Sl | Grouper serial list |
| Sx | Grouper serial list - expand open transactions |
| T | Grouper transaction statistics |

## The onstat -g nif Command

The **-g nif** option of the **onstat** command prints statistics about the network interface. The network interface information includes a summary of the number of buffers sent and received for each site.

The **-g nif** option is used primarily as a debugging tool and by Informix Technical Support. It can be helpful when data is not replicating.

The **onstat -g nif** command has the following formats:

```
onstat -g nif
onstat -g nif modifier
```

The modifier can have the values that the following table shows.

| Modifier | What the Command Prints |
|---|---|
| all | The sum and the sites |
| sites | The NIF site context blocks |
| serverid | Information about the replication server whose groupID is serverid |
| sum | The sum of the number of buffers sent and received for each site |

## The onstat -g que Command

The **-g que** option of the **onstat** command prints statistics for the high-level queue interface. The **-g que** option is used primarily as a debugging tool and by Informix Technical Support.

## The onstat -g rcv Command

The **-g rcv** option of the **onstat** command prints statistics about the receive manager. This option is used primarily as a debugging tool and by Informix Technical Support.

The **onstat -g rcv** command has the following formats:

```
onstat -g rcv
onstat -g rcv serverid
```

The serverid modifier causes the command to print only those output messages received from the replication server whose groupID is *serverid*.

## The onstat -g rep Command

The **-g rep** option of the **onstat** command prints events that are in the queue for the schedule manager. The **-g rep** option is used primarily as a debugging tool and by Informix Technical Support.

The **onstat -g rep** command has the following formats:

```
onstat -g rep
onstat -g rep replname
```

The replname modifier limits the output to those events originated by the replicate named replname.

## The onstat -g rqm Command

The **-g rqm** option of the **onstat** command prints statistics and / or contents of the low-level queues. If a queue is empty, no information is printed for that queue.

When you specify a modifier to select a specific queue, the command prints all the statistics for that queue and information about the first and last in-memory transactions for that queue.

The FULL modifier displays information about every in-memory transaction for every queue. The BRIEF modifier gives a brief summary of data in the queues and for which servers the data is queued. Use the BRIEF modifier to quickly identify sites where a problem exists. If large amounts of data are queued for a single server, then that server is probably down or off the network. The other modifiers of the **onstat -g rqm** command are used primarily as a debugging tool and by Informix Technical Support.

The **onstat -g rqm** command has the following formats:

```
onstat -g rqm
onstat -g rqm modifier
```

The modifier can have the values that the following table shows.

| Modifier | What the Command Prints |
|---|---|
| ACKQ | The ack Send Queue |
| CNTRLQ | The sync Send Queue |
| RECVQ | The Receive Queue |
| SENDQ | The Send Queue |
| SYNCQ | The sync Send Queue |
| FULL | Full information about every memory transaction for every queue |
| BRIEF | Brief summary of the data in the queues and the replication servers for which the data is queued. If a queue is empty, no information is printed for that queue. |

# Index


## A

Aborted-transaction spooling (ATS)
  activate 11-30, 11-36, 11-56
  change directory location 8-32
  deactivate 11-56, 12-26
  definition of 7-20
  description 10-3
  directory 11-35, 11-59
  field in structure 12-7
Active state, definition of 8-22
Add
  participant to replicate 8-14, 11-22, 12-32
  replicate to group 8-21, 11-20, 12-31
ANSI compliance, level Intro-16
API function
  cdr_connect() 12-10
  cdr_define_group() 12-11
  cdr_define_repl() 12-14
  cdr_define_serv() 12-18
  cdr_delete_group() 12-20
  cdr_delete_repl() 12-21
  cdr_delete_serv() 12-22
  cdr_error_reviewed() 12-23
  cdr_modify_group() 12-24
  cdr_modify_repl() 12-25
  cdr_modify_replmode() 12-27
  cdr_modify_serv() 12-28
  cdr_modify_servmode() 12-29
  cdr_move_errortab() 12-30
  cdr_participate_group 12-31
  cdr_participate_repl() 12-32
  cdr_prune_errors() 12-33
  cdr_prune_single_error() 12-34
  cdr_resume_group() 12-35
  cdr_resume_repl() 12-36
  cdr_resume_serv() 12-37
  cdr_start_cdr() 12-38
  cdr_start_group() 12-39
  cdr_start_repl() 12-40
  cdr_stop_cdr() 12-41
  cdr_stop_group() 12-42
  cdr_stop_repl() 12-43
  cdr_suspend_group() 12-44
  cdr_suspend_repl() 12-45
  cdr_suspend_serv() 12-46
  list of 12-8
  structure definitions 12-5
Arguments, in stored procedures 7-18
Asynchronous data replication, definition of 1-4, 1-5
ATS. *See* Aborted-transaction spooling.

## B

Begin work without replication 6-9
BLOB. *See* Simple large object and smart large object.
Blobspace 5-11
Boldface type Intro-8
BYTE data, distributing 5-14

## C

Canonical format 11-30, 12-16
Capacity planning
  for delete tables 5-7
  logical-log records 5-5
  spooling directories 5-7

Capture mechanisms, log-based
data capture 1-8
Cascading deletes, implementation
of 7-20
CDR_DSLOCKWAIT configuration
parameter A-2
CDR_EVALTHREADS
configuration parameter A-3
CDR_LOGDELTA configuration
parameter A-4
CDR_NIFCOMPRESS
configuration parameter A-5
CDR_NIFRETRY configuration
parameter A-7
CDR_QUEUEMEM configuration
parameter A-7
Central registry, sqlhosts E-2
Clock, synchronize time on 7-16
CLU. *See* Command-line utility.
Code set, ISO 8859-1 Intro-5
Code, sample, conventions for
Intro-12
Codeset conversion files 5-16
Collision, definition of 7-12
Column mapping
definition of 9-13
procedure D-5
Command-line conventions
elements of Intro-10
example diagram Intro-12
how to read Intro-12
Command-line utility
abbreviations 11-6
backslash 11-8
cdr change group 11-20
cdr change replicate 11-22
cdr connect server 11-24
cdr define group 11-25
cdr define replicate 11-27
cdr define server 11-33
cdr delete group 11-38
cdr delete replicate 11-38, 11-39
cdr delete server 11-40
cdr disconnect server 11-43
cdr error 11-44
cdr list group 11-47
cdr list replicate 11-48
cdr list server 11-50
cdr modify group 11-53
cdr modify replicate 11-55
cdr modify server 11-59
cdr resume group 11-61
cdr resume replicate 11-62
cdr resume server 11-63
cdr start 11-64
cdr start group 11-66
cdr start replicate 11-67
cdr stop 11-68
cdr stop group 11-70
cdr stop replicate 11-71
cdr suspend group 11-73
cdr suspend replicate 11-75
cdr suspend server 11-77
connect option 11-8, 11-12
error return codes 11-13
introduction 11-3
list of commands 11-5
participant modifier 11-11
participant syntax 11-9
terminology 11-3
Comment icons Intro-9
Complex transactions, evaluating
images D-1
Compliance, with industry
standards Intro-16
Configuration parameters 5-8
CDR_DSLOCKWAIT A-2
CDR_EVALTHREADS A-3
CDR_LOGDELTA A-4
CDR_NIFCOMPRESS A-5
CDR_NIFRETRY A-7
CDR_QUEUEMEM A-7
list of A-1
Conflict resolution
information 8-18
optimize behavior 11-29, 12-17
options 11-28
Conflict-resolution rule
ignore 7-20
stored procedure 7-17
time stamp 7-15
valid combinations 7-13
Conflict-resolution scope
row-by-row 7-14
transaction 2-16, 7-14
Connectivity information, sqlhosts.
*See* Sqlhosts information.
Constraint checking 7-14
Contact information Intro-16
Conventions, documentation
Intro-7

## D

Data delivery
resume 11-61, 11-62, 11-63, 12-35,
12-36, 12-37
suspend 11-73, 11-75, 11-77, 12-44,
12-45, 12-46
Data dissemination model 4-4
Data replication
asynchronous 1-4, 1-5
capture mechanisms
log-based data capture 1-8
trigger-based transaction
capture 1-6
key elements 1-8
synchronous, definition of 1-3
trigger-based data capture 1-6
Data types
BYTE and TEXT 5-11
floating point 5-11
SERIAL and SERIAL8 5-11
Database server
connect to 3-7, 3-24, 11-24
disconnect from 11-43
listing 11-50
modify attributes 11-59
register for Enterprise Replication
11-33, 12-18
remove from Enterprise
Replication 8-30, 11-40, 12-22
resume data delivery 11-63, 12-37
suspend data delivery 11-77, 12-46
Database server groups 3-4, 3-5, 8-6,
11-12, 12-3, E-4
Data-consolidation model 4-5
DB-Access utility Intro-5
DBSERVERALIASES 3-5, 11-12,
12-3, A-1
DBSERVERNAME 11-12, 12-3, A-1
dbspace
delete tables 5-7
error table 11-44, 12-30
message queues 5-6

receive queue 3-21, 11-35, 11-36, 12-19
send queue 11-35, 11-36, 12-19
Default locale Intro-5
Delete
database server from Enterprise Replication 11-40, 12-22
participant from replicate 11-22, 12-32
replicate 11-39, 12-21
replicate from group 11-20, 12-31
replicate group 11-38, 12-20
Delete table
definition of 5-7, 7-12
in conflict resolution 7-15, 7-18
Demonstration databases Intro-5
Dependencies, software Intro-4
directory 3-21
Documentation notes Intro-15
Documentation, types of
documentation notes Intro-15
error message files Intro-14
machine notes Intro-15
on-line help Intro-14
on-line manuals Intro-13
printed manuals Intro-14
related reading Intro-16
release notes Intro-15


## E


en_us.8859-1 locale Intro-5
Enterprise Replication threads, list of 5-8
Environment variables Intro-8
Error management 12-23, 12-30, 12-33, 12-34
Error message files Intro-14
Error messages, Enterprise Replication C-1
Error table
dbspace 11-44, 12-30
description 12-7
management 11-44, 12-33, 12-34
Expert mode, description of 8-33


## F


Feature icons Intro-9
Features of this product, new Intro-6
Find Error utility Intro-14
finderr utility Intro-14
Floating-point numbers, canonical format 11-30, 12-16
Frequency attributes
description 11-14
in list of replicates 11-48
of a replicate 11-58, 12-16, 12-25
of a replicate group 12-11, 12-12, 12-24
structure members 12-6


## G


Global Language Support (GLS)
locale Intro-5
locale of date 11-15
Group. *See* Replicate group, Database server groups.


## H


Help
context sensitive 8-37
menu 8-35
Hierarchical view
print schematic 8-38
save report 8-38
High-availability data replication, support of 5-3
Homogeneous platforms 5-15


## I


Icons
feature Intro-9
Important Intro-9
platform Intro-9
product Intro-9
Tip Intro-9
Warning Intro-9
Identifiers 8-8
Ignore conflict-resolution rule 7-20
Important paragraphs, icon for Intro-9
Inactive state, definition of 8-16, 8-22
Industry standards, compliance with Intro-16
INFORMIXDIR/bin directory Intro-5
INFORMIXSQLHOSTS environment variable E-2
ISO 8859-1 code set Intro-5


## L


List
Enterprise Replication servers 11-50
replicate groups 11-47
replicates 11-48
Locale
default Intro-5
en_us.8859-1 Intro-5
Locales
preparing 5-16
Log-based data capture 1-8
Logging, unbuffered 5-10
Logical log
capacity planning 5-5
reading of 7-3, 7-5
Long identifiers 8-8


## M


Machine notes Intro-15
Machine-independent format 11-30, 12-16
Menus
Help 8-35
Replicate 9-4
Replicate group 8-20
Server 8-25
View 8-32
Window 8-33
Message file for error messages Intro-14

Message queue
description 7-11
planning space for 5-6
*See also* Receive queue.
*See also* Send queue.
Mode
field in CDR_plist 12-5
of participant 11-55
set 11-10, 12-27, 12-29

## N

New features of this product Intro-6

## O

ONCONFIG file
configuration parameters A-1
setting DBSERVERNAME 11-12, 12-3, A-1
On-line help Intro-14
On-line manuals Intro-13
onstat command F-1
onunload and onload utilities, synchronizing data 6-3, 6-5
Operational limitations 5-3
Optical devices, not supported 5-11
Optimized mode, stored procedure 7-19

## P

Participant
adding to replicate 8-14, 9-15, 11-22, 12-32
change mode 12-27, 12-29
defining 12-14
definition of 2-11
deleting from replicate 8-14, 11-22, 12-32
owner option 8-29, 11-10
primary option 11-10
read-only option 11-10
syntax 11-9
Participant modifier
description 11-9
restrictions 5-15, 11-11
syntax 11-11
user-defined types 5-15
Platform icons Intro-9
Primary participant 8-17
Primary-target replication system 4-3
Printed manuals Intro-14
Processes
evaluating data for replication 7-11
evaluating distinct times 7-4
synchronizing data using conflict resolution 7-12
Product icons Intro-9
Program group
Documentation notes Intro-15
Release notes Intro-15
Properties
participant, viewing 8-19
replicate group, viewing replicate 8-24
replicate, viewing 8-17
server, viewing 8-31

## Q

Queue
receive, definition of 7-11
send, definition of 7-11
*See also* Receive queue.
*See also* Send queue.
Quiescent state
replicate, definition of 8-16, 8-23

## R

Receive queue
dbspace 12-19
definition of 7-11
field in structure 12-7
location 11-35, 11-36
planning space for 5-6
regedt32 program and sqlhosts registry E-3
Related reading Intro-16
Release notes Intro-15
Replicate
aborted-transaction spooling 11-30
adding participants 11-22, 12-32
adding to group 11-20
attributes, modify 12-25
change participant mode 12-27
conflict options 11-28
creating 11-27, 12-14
definition of 2-9
deleting 11-39, 12-21
deleting from group 11-20
deleting participants 11-22
listing 11-48
modify attributes 11-55
removing participants 12-32
resume data delivery 11-62
row-information spooling 11-30
start data capture 11-67
stop data capture 11-71
stop data delivery 12-43
suspend data delivery 11-75, 12-45
Replicate group
adding replicates 8-21
changing the state 8-22, 11-61
creating 8-20, 11-25, 12-11
definition of 2-12
deleting 8-21, 11-38, 12-20
listing 11-47
modify 11-53
modify attributes 12-24
performance 2-12
removing replicates 8-21
resume data delivery 11-61
start data capture 11-66
stop data capture 11-70
suspend data delivery 11-73, 12-44
Replicate states, how to change 8-14
Replication Event Monitor 8-11, 10-14
Replication Manager, toolbar icons 8-10

Replication order error, definition of 7-12
Replication server. *See* Database server.
Replication system
  primary-target 4-3
  update-anywhere 4-9
Replication volume, analyzing for capacity 5-10
Resume data delivery
  for a replicate 12-36
  for replicate group 12-35
  to a server 12-37
Return codes
  description 11-13
  list of C-1
RIS. *See* Row-information spooling.
rofferr utility Intro-14
Row-by-row, conflict-resolution scope 2-16, 7-14
Row-information spooling (RIS)
  activate 11-30, 11-36, 11-56
  deactivate 11-56, 12-26
  definition of 7-20
  directory 11-35, 11-59
  field in structure 12-7
  sample output 10-12
  syntax 10-9

## S

Safely stored, definition of 7-11
Sample-code conventions Intro-12
Scripting view 8-39
SELECT statement
  in syscdrpart B-9
  limitations 2-12, 11-11, 12-16
  performance 5-10
  with shadow columns 6-4
Semi-stable queue, definition B-14
Send queue
  dbspace 12-19
  definition of 7-11
  location 11-35, 11-36
  planning space for 5-6
SERIAL data type 4-10
Serial keys
  defining in update anywhere 4-9
Server group. *See* Database server group.
setup program, and sqlhosts registry E-1
Severe-error table. *See* Error table.
Shadow column
  information in ATS file 10-6
  resolving collisions 7-19
Shadow columns
  creating 6-4
  with BEGIN WORK WITHOUT REPLICATION 6-6
  with DB-Access 6-10
  with HPL 6-4
  with UNLOAD 6-6
Simple large objects 5-11
Software dependencies Intro-4
Spooling directories
  planning for capacity 5-7
  row information 7-20
  setting 3-21
Spooling information 8-19
SQL code Intro-12
SQL statements
  complex 5-10
  limitations 5-3
sqlhosts information
  INFORMIXSQLHOSTS environment variable E-2
  on UNIX 3-4, 8-6, 11-12, 12-4
  on Windows-NT 3-4, 8-6, E-2
Stable queue, use in distribution process 7-11
Start data delivery
  for a replicate 8-15
  for replicate 11-67, 12-40
  for replicate group 8-22, 11-66, 12-39
Start Enterprise Replication 11-64
Start replication 12-38
Status information, SMI tables B-1
Stop data delivery
  for a replicate 12-43
  for replicate group 12-42
Stop Enterprise Replication 11-68, 12-41
Stored procedure
  arguments 7-18
  example D-3
  SPL only 2-16
Stored-procedure conflict-resolution rule
  definition of 7-17
  evaluating blob data 5-13
stores_demo database Intro-5
superstores Intro-5
superstores_demo database Intro-5
Suspend data delivery
  for database server 12-46
  for replicate 12-45
  for replicate group 12-44
Suspend state, definition of 8-16, 8-23
Synchronization, of operating system times 5-9
Synchronize clocks 7-16
Synchronizing data
  using DB-Access 6-10
  using ESQL/C 6-10
  using onunload and onload utilities 6-5
Synchronous data replication, definition of 1-3
syscdr database, description of 11-17
System requirements
  database Intro-4
  software Intro-4
System-monitoring interface (SMI) tables
  list of B-1
  syscdrack_buf B-5
  syscdrack_txn B-6
  syscdrctrl_buf B-6
  syscdrctrl_txn B-6
  syscdrerror B-7, B-16
  syscdrgrp B-7
  syscdrgrppart B-8
  syscdrpart B-8
  syscdrprog B-9
  syscdrq B-10
  syscdrqueued B-11
  syscdrrecv_buf B-11

syscdrrecv_txn B-11
syscdrrepl B-12
syscdrs B-13
syscdrsend_buf B-14
syscdrsend_txn B-15
syscdrserver B-15
syscdrtx B-17
syscdrtxproc B-17

## T

Terms
  aborted-transaction spooling (ATS) 7-20
  row-information spooling (RIS) 7-20
TEXT data, distributing 5-14
Threads
  list of 5-8
  used by Enterprise Replication 5-8
Time synchronization 5-9
Time, evaluating 7-4
Time-stamp conflict-resolution rule
  definition of 7-15
  evaluating blob data 5-12
  synchronize clocks 7-16
Tip icons Intro-9
Transaction
  conflict-resolution scope 2-16, 7-14
  constraint checking 7-14
  evaluation examples 7-7–7-10
  evaluation logic 7-6
  processing, impact of 5-10
Trigger-based
  data capture 1-6
  transaction capture 1-6
Triggers
  activate firing 11-31, 11-56, 11-57
  deactivate firing 11-56, 12-26
  implementation of 7-21
  in multiple row images 7-10
  in primary key updates 7-10
  permissions on 11-10

## U

UDTs 5-15
UNIX operating system
  default locale for Intro-5
Update-anywhere replication system 4-9
User-defined data types 5-15

## V

Version number 8-37
Volume, analyzing replication volume 5-10

## W

Warning icons Intro-9
Windows NT
  default locale for Intro-5
Wizard
  define replicate 9-4
  define replicate group 9-15
  generic configuration 9-17
Workflow replication model 4-7
Workload-partitioning model 4-6

## X

X/Open compliance level Intro-16

## Symbols

$INFORMIXDIR/etc/sqlhosts. *See* sqlhosts file.